RedHawk™ Linux®

Frequency-Based Scheduler User's Guide

0898005-360
June 2017

Copyright 2017 by Concurrent Real-Time, Inc. All rights reserved.

本書は当社製品を利用する社員、顧客、エンドユーザーを対象とします。
本書に含まれる情報は、本書発行時点での正確な情報ですが、予告なく変更されることがあります。
当社は、明示的、暗示的に関わらず本書に含まれる情報に対して保障できかねます。

誤字・誤記の報告または本書の特定部分への意見は、当該ページをコピーし、コピーに修正またはコメントを記述してコンカレント日本株式会社まで郵送またはメールしてください。

http://www.concurrent-rt.co.jp/company/

本書はいかなる理由があろうとも当社の許可なく複製・変更することはできません。

Concurrent Real-Time, Inc.およびそのロゴはConcurrent Real-Time, Inc.の登録商標です。
当社のその他すべての製品名はConcurrent Real-Time, Inc.の商標です。また、その他全ての製品名が各々の所有者の商標または登録商標です。
Linux®は、Linux Mark Institute(LMI)のサブライセンスに従い使用しています。

改定履歴:

Revision History:	Level:	Effective With:
August 2002	000	RedHawk Linux Release 1.1 & 1.2
September 2003	110	RedHawk Linux Release 1.4
December 2003	200	RedHawk Linux Release 2.0
March 2004	210	RedHawk Linux Release 2.1
July 2004	220	RedHawk Linux Release 2.2
May 2005	230	RedHawk Linux Release 2.3
March 2006	240	RedHawk Linux Release 4.1
May 2007	250	RedHawk Linux Release 4.2
October 2007	260	RedHawk Linux Release 4.2
April 2008	300	RedHawk Linux Release 5.1
January 2009	320	RedHawk Linux Release 5.2.1
August 2009	330	RedHawk Linux Release 5.2.4
November 2009	340	RedHawk Linux Release 5.4
January 2013	350	RedHawk Linux Release 6.3
June 2017	360	RedHawk Linux Release 7.3

注意事項:

本書は、Concurrent Real-Time, Inc.より発行された「RedHawk Linux User's Guide」を日本語に翻訳した資料です。英文と表現が異なる文章については英文の内容が優先されます。

前書き

マニュアルの範囲

本書はRedHawk Linuxオペレーティング・システム上で動作するFrequency-Based Scheduler (FBS)およびPerformance Monitor (PM)について説明します。これらのユーティリティは、性能監視を目的として指定された頻度でプロセスをスケジュールするために使用されます。

このガイドにはreal-time command processor (**rtcp**)やCおよびFORTRANライブラリに関連する使用説明が含まれています。*NightSim RT User's Guide*に記述されていますので、FBSおよびPM用のグラフィカル・ユーザー・インターフェースであるNightSim™の使用説明については含まれていません。

マニュアルの構成

本書は以下のセクションで構成されます:

- 1章「*序文*」は、Frequency-Based SchedulerとPerformance Monitorだけでなくソフトウェア、構成、アクセス特権の必須要件の序論を提供します。
- 2章「*Frequency-Based Schedulerの使用*」は、Frequency-Based Scheduler、その機能、ユーザー・インターフェースの概要の説明を提供します。
- 3章「*タイミング・ソース*」は、Frequency-Based Scheduler用のタイミング・ソースとしてリアルタイム・クロック、エッジ・トリガー割込み、ユーザー提供デバイスを使用する方法を説明します。
- 4章「*Performance Monitorの使用*」は、Performance Monitor、その機能、ユーザー・インターフェースの概要の説明を提供します。
- 5章「*rtcpの使用*」は、real-time command processor (**rtcp**)を使用するための手順を説明して、そのコマンド各々に関する参考情報を提供します。
- 6章「*Cライブラリ・インターフェース*」は、Frequency-Based SchedulerとPerformance MonitorのCライブラリ・インターフェースを説明して、そのルーチン各々に関する参考情報を提供します。
- 7章「*FORTRANライブラリ・インターフェース*」は、Frequency-Based SchedulerとPerformance MonitorのFORTRANライブラリ・インターフェースを説明して、そのルーチン各々に関する参考情報を提供します。
- 付録A「*rtcpスクリプトの実例*」は、**rtcp**スクリプトの実例を含みます。
- 付録B「*rtcpエラー・メッセージ*」は、**rtcp**により報告される可能性のあるエラーの説明を提供します。
- 付録C「*Cインターフェースの実例*」は、Frequency-Based SchedulerとPerformance MonitorのCライブラリ・インターフェースの使い方を示すプログラム例を含みます。

「*用語解説*」は、本書に含まれている概念を理解するために重要となる技術的な用語の定義を含んでいます。

「*索引*」は、アルファベット順で重要な用語や概念とテキスト内に現れるページ数の参照を含みます。

構文記法

本書を通して使用される表記法は以下のとおりとなります。

斜体　ユーザーが特定する書類、参照カード、参照項目は、*斜体*にて表記します。特殊用語も*斜体*にて表記します。

太字　ユーザー入力は**太字**形式にて表記され、指示されたとおりに入力する必要があります。ディレクトリ名、ファイル名、コマンド、オプション、manページの引用も**太字**形式にて表記します。

`list`　プロンプト、メッセージ、ファイルやプログラムのリストのようなオペレーティング・システムおよびプログラムの出力は`list`形式にて表記します。

`[]`　ブラケット(大括弧)はコマンドオプションやオプションの引数を囲みます。もし、これらのオプションまたは引数を入力する場合、ブラケットをタイプする必要はありません。

ハイパーテキスト・リンク
本資料を見ている時に項、図、テーブル・ページ番号照会をクリックすると対応する本文を表示します。青字で提供されるインターネットURLをクリックするとWebブラウザを起動してそのWebサイトを表示します。赤字の出版名称および番号をクリックすると(アクセス可能であれば)対応するPDFのマニュアルを表示します。

関連図書

以下の表はConcurrent Real-Timeに関連するドキュメントの一覧です。これらのドキュメントは、Concurrent Real-TimeのWEBサイトにて参照または入手することが可能です。

http://redhawk.concurrent-rt.com/docs/

タイトル	Pub No.
RedHawk Linux Release Notes Version x.x	0898003
RedHawk Linux User's Guide	0898004
Real-Time Clock & Interrupt Module (RCIM) User's Guide	0898007
NightSim RT User's Guide	0890480
NightView RT User's Guide	0890395

x.x：リリース・バージョン

目次

前書き iii

1章 序文

Frequency-Based Schedulerの概要 1-1
Performance Monitorの概要 1-2
ソフトウェア要件 1-2
構成 1-3
Frequency-Based Scheduler 1-3
Performance Monitor 1-3
特権 1-4

2章 Frequency-Based Schedulerの使用

Frequency-Based Schedulerとは何か？ 2-1
スケジューラーの頻度はどのように定義されるのか？ 2-2
プロセスはどのようにスケジュールされるのか？ 2-2
フレーム・オーバーランの許容 2-4
デッドライン違反の検出 2-5
ユーザー・インターフェース 2-7
Frequency-Based Schedulerプロセスのデバッグ 2-7

3章 タイミング・ソース

概要 3-1
リアルタイム・クロックの使用 3-1
リアルタイム・クロック・デバイスの理解 3-1
ユーザー・インターフェースの理解 3-2
リアルタイム・クロックの使用に関する基本手順 3-2
エッジ・トリガー割込みの使用 3-3
エッジ・トリガー割込みの理解 3-3
ユーザー・インターフェースの理解 3-4
ユーザー提供タイミング・デバイスの使用 3-4

4章 Performance Monitorの使用

Performance Monitorとは何か? 4-1
何の値を監視するのか? 4-1
アイドル時間の監視 4-3
非スケジュール・プロセスの監視 4-4
ユーザー・インターフェース 4-5
シミュレーション性能の最適化 4-5

5章 rtcpの使用

リアルタイム・コマンド・プロセッサーとは何か? 5-1
rtcpとFrequency-Based Scheduler 5-1

rtcpとPerformance Monitor 5-2
実行モード 5-2
ダイレクト・モードの使用 5-2
コマンド名と引数を使ったrtcpの起動 5-3
スクリプト・ファイルからリダイレクトされたコマンドを使ったrtcpの起動. . . 5-3
rtcpを呼び出すスクリプトの起動 5-3
対話モードの使用 5-4
困ったときは 5-5
rtcpコマンド 5-7
rtcpコマンドの要約 5-7
コマンドの順序 5-8
rtcpコマンドの使用 5-10
ats – Frequency-Based Schedulerにタイミング・ソースを接続 5-11
chs – Frequency-Based Schedulerのアクセス権を変更 5-12
cs – Frequency-Based Schedulerを構成 5-13
dts – Frequency-Based Schedulerからタイミング・ソースを分離 5-15
rms – Frequency-Based Schedulerを削除 5-16
svs – スケジューラの構成を保存 5-17
vc – マイナー・サイクル/メジャー・フレームの総数を表示 5-18
vs – スケジューラーの構成を表示 5-19
ls – システム上の全スケジューラーを表示 5-21
rc – リアルタイム・クロックを開始 5-21
sc – リアルタイム・クロックを停止 5-22
stc – リアルタイム・クロックを設定 5-22
gtc – リアルタイム・クロックの設定を表示 5-23
start – Frequency-Based Schedulerのスケジューリングを開始 5-24
resume – Frequency-Based Schedulerのスケジューリングを再開 5-25
stop – Frequency-Based Schedulerのスケジューリングを停止. 5-25
rmp – Frequency-Based Schedulerからプロセスを削除 5-26
rsp – プロセスを再スケジュール 5-28
sp – Frequency-Based Schedulerのプロセスをスケジュール 5-32
vp – Frequency-Based Schedulerのプロセスを表示 5-35
cpm – Performance Monitorの値を消去 5-38
pm – Performance Monitorを開始/停止 5-40
vcm – Performance Monitorのタイミング・モードを表示/修正 5-42
vpm – Performance Monitorの値を表示 5-43
ex – リアルタイム・コマンド・プロセッサーを終了 5-47
he – ヘルプ情報を表示 5-47

6章 Cライブラリ・インターフェース

概要 6-1
プログラムのコンパイルとリンク. 6-1
Big-SMP FBSインターフェース 6-2
Frequency-Based Schedulerルーチン 6-4
ルーチンの要約 6-4
Cライブラリ・コールの順序 6-6
Frequency-Based Schedulerルーチンの使用 6-7
fbsaccess – Frequency-Based Schedulerのアクセス権を変更 6-7
fbsattach – Frequency-Based Schedulerにタイミング・ソースを接続 6-8
fbsavail – Frequency-Based Schedulerが構成されているかどうかを問合せ 6-9
fbsconfigure – Frequency-Based Schedulerを構成 6-10
fbscycle – マイナー・サイクル/メジャー・フレームの総数を返答 6-13

fbsdetach – Frequency-Based Schedulerからタイミング・ソースを分離 6-14
fbsdir – スケジューラー・キーのリストを返答 6-15
fbsgetpid – スケジュール済みプロセスのプロセスIDを返答 6-16
fbsgetrtc – リアルタイム・クロックの現在値を取得 6-17
fbsid – キーに対するFrequency-Based Scheduler識別子を返答 6-18
fbsinfo, fbsinfo_big – Frequency-Based Schedulerの情報を返答 6-19
fbsintrpt – スケジューリングを開始/停止/再開 6-22
fbsremove – Frequency-Based Schedulerを削除 6-23
fbsresume – Frequency-Based Schedulerのスケジューリングを再開 6-24
fbsrunrtc – リアルタイム・クロックを開始/停止 6-25
fbsschedself – Frequency-Based Schedulerに呼び出し元プロセスを追加 6-26
fbssetrtc – リアルタイム・クロックを設定 6-28
fbstrig – スリープ中のFrequency-Based Schedulerプロセスを実行可能に遷移 ... 6-29
fbswait – Frequency-Based Schedulerを待機 6-30
nametopid, namepid, nametopid_big, namepid_big – 指定したプロセス名称のプロセスIDを返答 6-31
pgmremove, pgmremove_big – Frequency-Based Schedulerからプロセスを削除 ... 6-33
pgmtrigger – Frequency-Based Schedulerのプロセスを起動 6-35
sched_fbsqry, sched_fbsqry_big – Frequency-Based Schedulerのプロセスを問合せ . 6-36
sched_pgm_deadline_query, sched_pgm_deadline_query_big – プロセスに割り当て済みデッドライン時間を問合せ 6-39
sched_pgm_deadline_test, sched_pgm_deadline_test_big – デッドライン違反の有無をテスト 6-42
sched_pgm_set_deadline, sched_pgm_set_deadline_big – プロセス・デッドライン時間を設定または消去 6-45
sched_pgm_set_soft_overrun_limit, sched_pgm_set_soft_overrun_limit_big – ソフト・オーバーラン上限を設定 6-48
sched_pgmadd, sched_pgmadd_big – Frequency-Based Schedulerにプロセスをスケジュール 6-51
sched_pgmadd_args, sched_pgmadd_args_big– Frequency-Based Schedulerに引数を付けてプロセスをスケジュール 6-54
sched_pgmadd_attr, sched_pgmadd_attr_big – Frequency-Based Schedulerに引数と属性を付けてプロセスをスケジュール 6-58
sched_pgmqry, sched_pgmqry_big – プロセスを問合せ 6-61
sched_pgmresched, sched_pgmresched_big– プロセスを再スケジュール 6-65
Performance Monitorルーチン 6-69
ルーチンの要約 6-69
Cライブラリ・コールの順序 6-69
Performance Monitorルーチンの使用 6-70
pmclrpgm, pmclrpgm_big, – プロセスの値を消去 6-71
pmclrtable, pmclrtable_big – プロセッサーの値を消去 6-73
pmmonitor, pmmonitor_big – プロセッサーのPerformance Monitorを開始/停止 ... 6-75
pmprogram, pmprogram_big – プロセスのPerformance Monitorを開始/停止 6-77
pmqrycpu, pmqrycpu_big – 選択したプロセッサーの値を問合せ 6-79
pmqrylist – プロセスのリストの値を問合せ 6-83
pmqrypgm, pmqrypgm_big – 選択したプロセスの値を問合せ 6-85
pmqrytimer – Performance Monitorのモードを問合せ 6-88
pmselect – Performance Monitorのモードを選択 6-89

7章 FORTANライブラリ・インターフェース

概要 7-1
プログラムのコンパイルとリンク 7-1
Big-SMP FBSインターフェース 7-2
Frequency-Based Schedulerルーチン 7-3
ルーチンの要約 7-3
FORTRANライブラリ・コールの順序 7-5
Frequency-Based Schedulerルーチンの使用 7-6
fbsaccess – Frequency-Based Schedulerのアクセス権を変更 7-6
fbsattach – Frequency-Based Schedulerにタイミング・ソースを接続 7-8
fbsconfigure – Frequency-Based Schedulerを構成 7-9
fbscycle – マイナー・サイクル/メジャー・フレームの総数を返答 7-11
fbsdetach – Frequency-Based Schedulerからタイミング・ソースを分離 7-12
fbsgetrtc – リアルタイム・クロックの現在値を取得 7-13
fbsid – キーに対するFrequency-Based Scheduler識別子を返答 7-14
fbsinfo – Frequency-Based Schedulerの情報を返答 7-15
fbsintrpt – スケジューリングを開始/停止/再開 7-17
fbsquery – Frequency-Based Schedulerのプロセスを問合せ 7-18
fbsremove – Frequency-Based Schedulerを削除 7-21
fbsresume – Frequency-Based Schedulerのスケジューリングを再開 7-22
fbsrunrtc – リアルタイム・クロックを開始/停止 7-24
fbsschedself – Frequency-Based Schedulerにプロセス/スレッドをスケジュール . . . 7-25
fbssetrtc – リアルタイム・クロックを設定 7-27
fbswait – Frequency-Based Schedulerを待機 7-28
nametopid – 指定したプロセス名称のプロセスIDを返答 7-29
pgmquery – Frequency-Based Schedulerのプロセスを問合せ 7-30
pgmremove – Frequency-Based Schedulerからプロセスを削除 7-32
pgmreschedule – プロセスを再スケジュール 7-34
pgmschedule – Frequency-Based Schedulerにプロセスをスケジュール 7-37
pgmstat – FBSにスケジュールされたプロセスの状態を問合せ 7-40
pgmtrigger – FBSの待ち状態プロセスを起動 7-42
rtparm – 初期パラメータを返答 7-43
sched_pgm_deadline_query – プロセスに割り当て済みデッドラインを問合せ . . . 7-44
sched_pgm_deadline_test – デッドライン違反の有無をテスト 7-46
sched_pgm_set_deadline – プロセス・デッドライン時間を設定または消去 7-48
sched_pgm_set_soft_overrun_limit – ソフト・オーバーラン上限を設定 7-50
sched_pgm_soft_overrun_query – ソフト・オーバーラン処理を問合せ 7-51
schedfbsqry – Frequency-Based Schedulerのプロセスを問合せ 7-52
schedpgmadd – Frequency-Based Schedulerのプロセスをスケジュール 7-55
schedpgmadd_args – Frequency-Based Schedulerに引数を付けてプロセスをスケジュール 7-57
schedpgmqry – Frequency-Based Schedulerのプロセスを問合せ 7-59
schedpgmresched – プロセスを再スケジュール 7-62
Performance Monitorルーチン 7-65
ルーチンの要約 7-65
FORTRANライブラリ・コールの順序 7-65
Performance Monitorルーチンの使用 7-67
pmclrpgm, – プロセスの値を消去 7-67
pmclrtable – プロセッサーの値を消去 7-69
pmmonitor – プロセッサーのPerformance Monitorを開始/停止 7-70
pmprogram – プロセスのPerformance Monitorを開始/停止 7-71
pmqrycpu – 選択したプロセッサーの値を問合せ 7-73
pmqrylist – プロセスのリストの値を問合せ 7-75

pmqrypgm – 選択したプロセスの値を問合せ 7-77
pmquerytimer – Performance Monitorのモードを問合せ 7-80
pmselect – Performance Monitorのモードを選択 7-81

付録A rtcpスクリプトの実例 A-1

付録B rtcpエラー・メッセージ B-1

付録C Cインターフェースの実例 C-1

schedule.c. C-2
prog.c. C-8
Makefile C-8

用語解説 Glossary-1

索引 Index-1

画面

画面5-1. rtcpコマンドの表示 5-5
画面5-2. rtcp引数の初期画面 5-6
画面5-3. rtcp引数の2番目の画面 5-7

図

図5-1. rtcpコマンドの順序： Frequency-Based Scheduler 5-9
図5-2. rtcpコマンドの順序： Performance Monitor 5-10
図6-1. Cライブラリ・コールの順序： Frequency-Based Scheduler 6-6
図6-2. Cライブラリ・コールの順序： Performance Monitor 6-70
図7-1. FORTRANライブラリ・コールの順序： Frequency-Based Scheduler 7-5
図7-2. FORTRANライブラリ・コールの順序： Performance Monitor 7-66

表

表2-1. プロセス・スケジューリング例 1 2-3
表2-2. プロセス・スケジューリング例 2 2-4
表3-1. RTCを使用するためのrtcpコマンドとルーチン 3-3
表3-2. ETIを使用するためのrtcpコマンドとルーチン 3-4
表5-1. rtcpコマンド 5-7
表6-1. Frequency-Based SchedulerのCライブラリ・ルーチン 6-4
表6-2. Frequency-Based Schedulerのアクセス権 6-7
表6-3. Performance MonitorのCライブラリ・ルーチン 6-69
表7-1. Frequency-Based SchedulerのFORTRANライブラリ・ルーチン 7-3
表7-2. Frequency-Based Schedulerのアクセス権 7-6
表7-3. 配列エレメントの中身: fbsinfo 7-15
表7-4. 配列エレメントの中身: fbsquery 7-19
表7-5. 配列エレメントの中身: fbsschedself 7-26
表7-6. 配列エレメントの中身: schedfbsqry 7-53
表7-7. Performance MonitorのFORTRANライブラリ・ルーチン 7-65

表7-8. 配列エレメントの中身: pmqrycpu 7-74
表7-9. 配列エレメントの中身: pmqrylis 7-76
表B-1. rtcpエラー B-1

1章

序文

Frequency-Based Schedulerの概要 1-1
Performance Monitorの概要 1-2
ソフトウェア要件 1-2
構成 1-3
    Frequency-Based Scheduler 1-3
    Performance Monitor 1-3
特権 1-4

1章

序文

本章では Frequency-Based Scheduler (FBS)と Performance Monitor (PM)を案内します。また、ソフトウェアと構成必要条件を提供し、これらのユーティリティを使用するための特権やケーパビリティについて説明します。

Frequency-Based Schedulerの概要

Frequency-Based Scheduler (FBS)は、指定された頻度でプロセスを起動するために使用するタスク同期メカニズムです。頻度は、例えば Real-Time Clock and Interrupt Module (RCIM)より提供される高分解能クロック、外部割込みソース、サイクルの終了をベースにすることが可能です。プロセスは優先度ベースのスケジューラーを使ってスケジュールされます。FBS は複数の周期的実行、オーバーラン検出しながらメジャーおよびマイナー・サイクルを利用する同調的なプロセス群を制御することが可能です。

Performance Monitor と一体で使用した場合、FBS は特定のアプリケーションの様々なタスクにプロセッサーを割り当てる最善の方法を決定するために使用することが可能です。

特定のアプリケーションの要求を満たすために FBS を簡単に構成することが可能です。複数の Frequency-Based Scheduler を異なる目的のために様々なスケジューリング手法を許可する異なるケーパビリティをそれぞれ使って構成することが可能です。Frequency-Based Scheduler の詳細は 2 章で提供します。

FBS のスケジューリングに関連する主要な機能へのアクセスは、次のインターフェースを通して実現します：

- **`rtcp`** (real-time command processor)は5章で説明します。
- C言語で書かれたアプリケーション・プログラムから呼び出し可能なライブラリ・ルーチン一式。このライブラリ・ルーチンは6章で説明します。
- FORTRANで書かれたアプリケーション・プログラムから呼び出し可能なライブラリ・ルーチン一式。このライブラリ・ルーチンは7章で説明します。
- FBSのグラフィカル・ユーザー・インターフェース(GUI)であるNightSim。NightSimは *NightSim RT User's Guide*で説明します。

Performance Monitorの概要

Performance Monitor (PM)は Frequency-Based Scheduler でスケジュールされたプロセスによるCPU の使用を監視します。取得した値は、負荷バランスや処理効率を向上させるためにプロセス群をプロセッサー間でどのように再分配するかを判定するのを支援します。Performance Monitor とそのケーパビリティの詳細は 4 章で提供します。

Performance Monitor に関連する主要な機能へのアクセスは、次のインターフェースを通して実現します：

- **`rtcp`** (real-time command processor)は5章で説明します。
- C言語で書かれたアプリケーション・プログラムから呼び出し可能なライブラリ・ルーチン一式。このライブラリ・ルーチンは6章で説明します。
- FORTRANで書かれたアプリケーション・プログラムから呼び出し可能なライブラリ・ルーチン一式。このライブラリ・ルーチンは7章で説明します。
- PMのグラフィカル・ユーザー・インターフェース(GUI)であるNightSim。NightSimは *NightSim RT User's Guide*で説明します。

ソフトウェア要件

RedHawk Linux オペレーティング・システムと共に配布される Frequency-Based Scheduler (FBS) RPM は、Frequency-Based Scheduler、Performance Monitor、**`rtcp(1)`** のためのカーネル・サポートを提供します。FBS と PM にアクセスするにはこのソフトウェアをインストールする必要があります。インストール手順に関しては *RedHawk Linux Release Notes* の適切なバージョンを参照して下さい。

FBS プロセスのタイミング用に RCIM デバイスを使用する場合は、Real-Time Clock and Interrupt Module (RCIM)がシステムに組み込まれて構成されている必要があります。インストール手順に関しては *RedHawk Linux Release Notes* の適切なバージョンを参照して下さい。RCIM ドライバーの構成およびクロックと割り込みの利用に関する情報については、*Real-Time Clock and Interrupt Module (RCIM) User's Guide* を参照して下さい。

構成

Frequency-Based Scheduler および Performance Monitor に関する構成必要条件を以下に記載します。カーネル・パラメータの変更方法については *RedHawk Linux User's Guide* の「カーネルの構成および構築」章を参照して下さい。

Frequency-Based Scheduler

以下のシステム調整パラメータは FBS の動作に影響します。

FBSCHED	カーネル構成GUIの**`Frequency-Based Scheduler`**項を通してアクセス可能な本パラメータは、FBSを構成します。デフォルトでFBSは各々のRedHawk Linuxプレビルト・カーネルに設定されています。必要に応じてモジュールとして構成することが可能です。
RCIM_IRQ_EXTENSIONS	カーネル構成GUIの**`Device Drivers`**項を通してアクセス可能な本パラメータは、FBSを使用するために設計されたRCIMドライバーの特殊なサービス用フックを提供します。

Performance Monitor

カーネルに FBS を構成することに加えて、Performance Monitor は以下を必要とします：

FBSCHED_PM	カーネル構成GUIの**`Frequency-Based Scheduler`**項を通してアクセス可能な本調整パラメータは、システムにPerformance Monitorを構成するために使用されます。各々のRedHawk Linuxプレビルト・カーネルにデフォルトで設定されています。
HRACCT	カーネル構成GUIの**`General Setup`**項を通してアクセス可能な本パラメータは、Performance Monitorを使用するために必要となります。これは高分解能プロセス・アカウンティングを有効にして、タイミング値に関する各プロセスの実行時間を計測する手段を提供します。本パラメータはRedHawk Linuxのdebugとtraceカーネルにデフォルトで構成されています。

特権

RedHawk Linux はロール(役割)・ベースのアクセス制御方式を提供する **pam_capability(8)** と呼ばれる Pluggable Authenication Module (PAM)をサポートします。この方式では、**capability.conf** ファイル内の一連のロールを設定します。ロールは有効な Linux ケーパビリティのセットとして定義されます。

RedHawk システムは「fbscheduser」ロールが構成されています。このロールは FBS サブシステムを構成するライブラリやカーネル・コードを実行するために必要となる一連のケーパビリティを定義します。FBS ユーザーはデフォルトで「fbscheduser」ロールにより提供されるケーパビリティを(最小限でも)使用するには構成する必要があります。これは「fbscheduser」ロールが定義される各 FBS ユーザーに対して**/etc/security/capability.conf** に 1 行を追加することで成立します：

```
user      username          fbscheduser
```

username はユーザーの名前になります。

Frequency-Based Scheduler 上でスケジュールされたプログラムは、それらが利用するシステム・コールを行うことが出来る他のケーパビリティを必要とする可能性があります。fbscheduser ロールはより汎用的なロールを生成するための構成要素として使用される可能性があります。例えば、以下の 2 つのエントリーは「software」という名前のグループ内の全ユーザーに FBS サブシステム(fbscheduser)を使用するためのケーパビリティ、**mlock(2)** と **mlockall(2)** (cap_ipc_lock)を使用するためのケーパビリティに加えてユーザー定義のロール(user_defined)の他のケーパビリティを許可することになります。

```
role     realtimeuser    fbscheduser     cap_ipc_lock user_defined
group    software        realtimeuser
```

システム初期化中、(デフォルトで)RCIM デバイス・ファイルは全ユーザーが書き込み可能(モード： 0666)で生成されます。管理者が特定のユーザーだけにアクセス権を制限することを望む場合、それらのファイルを書き込むためのアクセス権を FBS ユーザーに与えるようにする必要があります。それを実現する 1 つの方法は RCIM デバイス・ファイルへ書き込みアクセスを提供するグループを定義し、FBS ユーザーをそのグループに割り当てることです。それを行う方法の例が次となります。

「realtimer」という名前のグループを追加するには、次を行ってください：

```
# groupadd -r realtimer
```

次に**/etc/sysconfig/rcim** を以下のように編集し設定して raltimer グループに RCIM デバイス・ファイルへの書き込みアクセスを与えて下さい：

```
RCIM_OWN=root
RCIM_GRP=realtimer
RCIM_PERM=0664
```

最後に FBS ユーザーをグループに追加して下さい。この例では「joe」を realtimer グループに追加します：

```
# adduser -u 50 -g 100 -d /home/joe -G realtimer -s /bin/bash -c
''Joe User'' joe
```

2章

Frequency-Based Schedulerの使用

Frequency-Based Schedulerとは何か？ 2-1
スケジューラーの頻度はどのように定義されるのか？. 2-2
プロセスはどのようにスケジュールされるのか？ 2-2
フレーム・オーバーランの許容 2-4
デッドライン違反の検出 2-5
ユーザー・インターフェース 2-7
Frequency-Based Schedulerプロセスのデバッグ 2-7

2章

Frequency-Based Schedulerの使用

本章は Frequency-Based Scheduler とその機能の説明を含みます。どのようにスケジューラーの頻度を定義するのか、どのようにプロセスをスケジュールするのか、オーバーランとは何か、どのようにこれらを管理するのかを説明します。スケジューラーにアクセスするためのユーザー・インターフェースを紹介します。

Frequency-Based Schedulerとは何か？

Frequency-Based Scheduler (FBS)は、指定した頻度と(オプションで)指定したデッドラインでプロセスを実行することを可能にするタスク同期メカニズムです。複数の Frequency-Based Scheduler を構成して同時に使用することが可能です。スケジューリングの頻度はユーザー定義のメジャー・フレームを構成するマイナー・サイクルの数がベースになります。各スケジューラーは、スケジューラの頻度を定義するためにいくつかの利用可能なタイミング・ソースのいずれかを使用する、またはサイクルの終了時にスケジュールすることが可能です。更にフレームのオーバーランを監視することも可能です。

具体的には FBS は以下を提供します：

- マイナー・サイクルの持続時間とメージャー・フレームあたりのマイナー・サイクル数を単位として頻度を定義
- プロセスをスケジュールするスケジューリング・パラメータを指定
- 1つのプロセッサーから全てのスケジューリング機能を制御(つまり、任意のプロセッサ上にプロセスをスケジュールおよび問合せ)
- デッドライン時間が割り当てられたFBSでスケジュール済みプロセスのデッドライン違反を検出
- FBSでスケジュールされた全てのプロセスのフレーム・オーバーランを検出
- FBSでスケジュールされた1つのプロセス、1つのプロセッサー上のFBSでスケジュールされた全てのプロセス、全てのプロセッサー上のFBSでスケジュールされた全てのプロセスのステータスを取得
- スケジューラーからFBSでスケジュールされた1つまたは全てのプロセスを削除
- FBSでスケジュールされた全てのプロセスを再スケジュール
- Frequency-Based Schedulerのスケジューリングを開始、停止、再開
- タイミング・ソースをFrequency-Based Schedulerに接続およびFrequency-Based Schedulerから切断
- Frequency-Based Schedulerのタイミング・ソースとしてリアルタイム・クロック・デバイスの使用を制御
- 単一プロセッサーまたはマルチプロセッサー環境においてシステム全体で最大100個のFrequency-Based Schedulerを構成
- FBSのスケジューリングと静的優先度のスケジューリングの両方を同時に使用

- FBSでスケジュールされたプロセスに対してソフト・オーバーラン上限を設定
- ソフト・オーバーラン上限およびFBSでスケジュールされたプロセスで被ったソフト・オーバーランの総数を問合せ

スケジューラーの頻度はどのように定義されるのか？

スケジューラーの頻度は、**`fbsconfigure(3)`**ライブラリ・ルーチンまたは **`rtcp(1)`**のいずれかを使って*メジャー・フレーム*を構成する*マイナー・サイクル*の数を定義することである程度定義されます。

Frequency-Based Scheduler によって保持される頻度の最小単位として、マイナー・サイクルは持続時間をそれと関連付けます。持続時間とはスケジューラーに加えられたタイミング・ソースによって生成される割り込みの合間を経過する時間です。例えば、タイミング・ソースがリアルタイム・クロックである場合、マイナー・サイクルの持続時間はマイナー・サイクルあたりのクロック・カウント数とクロック・カウント当たりのマイクロ秒数を指定することで定義されます。

メジャー・フレームとは、Frequency-Based Scheduler が構成されている全てのマイナー・サイクルを 1 回通り抜けることです。メジャー・フレームの持続時間はマイナー・サイクルの持続時間にメジャー・フレームあたりのマイナー・サイクル数を掛けることで決まります。

例えば、メジャー・フレームあたり 100 個のマイナー・サイクルを持つスケジューラーを構成し、タイミング・ソースとして 10,000 のクロック・カウントとクロック・カウントの持続時間が 1 マイクロ秒を備えるリアルタイム・クロックを使用する場合、各マイナー・サイクルは 10,000 マイクロ秒または 0.01 秒の持続時間で各フレームは 1 秒の持続時間となります。

*エンド・オブ・サイクル・スケジューリング*は現在のメジャー・フレームの現在のマイナー・サイクル中にスケジュールされた最後のプロセスがその処理を終えると実行されます。

プロセスはどのようにスケジュールされるのか？

プロセスが各メジャー・フレームで起動される最初のマイナー・サイクル(*開始ベース・サイクル*と呼ぶ)とそれが起動される頻度(*周期*と呼ぶ)を指定することで、プロセスは特定の頻度で実行するようにスケジュールされます。

例えば、開始ベース・サイクルを 0 および周期を 2 で Process-1 をスケジュールした場合、そのプロセスはフレーム内の最初のマイナー・サイクルで開始され、2 マイナー・サイクルごとに 1 回起動されます。

開始ベース・サイクルを 1 および周期を 4 で Process-2 をスケジュールした場合、そのプロセスはフレーム内の 2 番目のマイナー・サイクルで開始され、4 マイナー・サイクルごとに 1 回起動されます。

開始ベース・サイクルを 2 および周期を 2 で Process-3 をスケジュールした場合、そのプロセスはフレーム内の 3 番目のマイナー・サイクルで開始され、2 マイナー・サイクルごとに 1 回起動されます。

メジャー・フレームあたり 100 個のマイナー・サイクルで構成されたスケジューラーにおいて、これらのプロセスは表 2-1 で示すように起動します。

表2-1 プロセス・スケジューリング例 1

マイナー・サイクル	起動するプロセス
0	Process-1
1	Process-2
2	Process-1, Process-3
3	
4	Process-1, Process-3
5	Process-2
.	
.	
.	
97	Process-2
98	Process-1, Process-3
99	

プロセスがスケジュール可能な最大頻度はマイナー・サイクルあたり 1 回(周期は 1)で、最小頻度はメジャー・フレームあたり 1 回(この例では周期は 100)です。

プロセスはスリープさせるための FBS ライブラリ・ルーチンを呼び出すまで実行します。FBS は現在のメジャー・フレームの現在のマイナー・サイクルで起動するようにスケジュールされたこれらのスリープ中のプロセスを起こし、現在のフレーム内の各マイナー・サイクルでこの処理を繰り返します。スケジューラーが無効になるまで全てのメジャー・フレームにおいて処理全体を繰り返し続けます。

前述の例を拡張するため、以前の例から 3 つのプロセスを使用します :

- Process-1：開始ベース・サイクル=0、周期=2
- Process-2：開始ベース・サイクル=1、周期=4
- Process-3：開始ベース・サイクル=2、周期=2

同様にメジャー・フレームあたり 100 個のマイナー・サイクル、マイナー・サイクルの持続時間が 10,000 マイクロ秒(0.01 秒)、メジャー・フレームの持続時間が 1 秒のスケジューラを構成する場合、3 つのプロセスは表 2-2 で示すようにスケジュールされます。

表2-2 プロセス・スケジューリング例 2

メジャー・フレーム	時間 (秒)	マイナー・サイクル	起動するプロセス
0	0	0	Process-1
	0.01	1	Process-2
	0.02	2	Process-1, Process-3
	…		
	0.97	97	Process-2
	0.98	98	Process-1, Process-3
	0.99	99	
1	1.00	0	Process-1
	1.01	1	Process-2
	1.02	2	Process-1, Process-3
	…		
	1.97	97	Process-2
	1.98	98	Process-1, Process-3
	1.99	99	
	…		
n	*n*.00	0	Process-1
	n.01	1	Process-2
	n.02	2	Process-1, Process-3
	…		
	n.97	97	Process-2
	n.98	98	Process-1, Process-3
	n.99	99	

表 2-2 に示すように現在のメジャー・フレームがゼロで現在のマイナー・サイクルもゼロの場合、スケジューラーは Process-1 を起こします。0.01 秒後に Process-2 を起こして 0.02 秒後に Process-1 と Process-3 を起こします。

1 秒後、再度現在のメジャー・フレームがゼロになって現在のマイナー・サイクルもゼロになる場合、スケジューラーは Process-1 を起こします。0.01 秒後に Process-2 を起こして 0.02 秒後に Process-1 と Process-3 を起こします。スケジューラーは有効である限りこの処理を繰り返し続けます。

フレーム・オーバーランの許容

*フレーム・オーバーラン*はスケジュールされたプロセスが再度実行するようにスケジュールされる前にその処理が終了しない場合に発生します。

フレーム・オーバーランは 2 つのカテゴリーに分類されます：

hard overruns	スケジュールされたプロセスの致命的な障害
soft overruns	プロセスが許容されたソフト・オーバーラン数の上限に達した場合のみ致命的な障害とする。スケジュールされた各プロセスはソフト・オーバーラン上限があり、デフォルトは0です。 プロセスを妥当なソフト・オーバーラン数で存続させることはシステムをより柔軟かつ効率的にします。一部のソフト・オーバーランは無作為、予測不可能、または再発する可能性の低い外部イベントに起因します。その他のソフト・オーバーランはマイナー・フレーム・オーバーランに起因します。ソフト・オーバーランはスケジュールされたプロセスをフレーム・オーバーランから回復し同期に復帰する機会を与えます。

エンド・オブ・サイクル・スケジューリングに関連するオーバーランは存在しないことに注意して下さい。定義上、スケジュールされた全てのプロセスが **`fbswait(3)`** コールでブロックされた時にサイクルは終了します。

RedHawk Linux はスケジュールされた各プロセスに関してソフトおよびハード・オーバーランの両方をカウントしていますが、各スケジューラーに関してはハード・オーバーランだけとなります。他のプロセスはスケジュールされたプロセスまたはスケジューラーに問い合わせることでそれらの総数を取得することが可能です。

プロセスをスケジュールする時、その下で実行中のプロセスがハード・オーバーランを引き起こした場合にスケジューラーをカーネルにより停止させることを指定することが可能です。プロセスがハード・オーバーランを引き起こした場合にスケジューラーを停止することを指定していないとハード・オーバーランの数に関わらずスケジューラーは実行し続けます。

プロセスをスケジュールする時、プロセスが許容しカーネルによりソフト・オーバーランとして処理された連続的なソフト・オーバーラン上限総数を指定することが可能です。デフォルトの連続的なソフト・オーバーラン上限はゼロであることに注意して下さい。上限をゼロに設定するとプロセスが招く*全*てのオーバーランはハード・オーバーランとして処理されます(下記参照)。

フレーム割り込みが発生した時､スケジュールされたプロセスがオーバーランし **`fbswait(3)`** でブロックするように構成されていない場合、カーネルはこのオーバーランを以下の手順によりソフトまたはハード・オーバーランとして処理するかどうかを決定します：

- そのプロセスの連続的なソフト・オーバーラン総数が増加します。
- 総数がプロセスのソフト・オーバーラン上限に到達しないまたは超えない場合、オーバーランはソフト・オーバーランとして処理されます。プロセスは次回**`fbswait`**をコールした時にブロックしませんが、その代わりにステータス値が2となって**`fbswait`**コールから直ちに戻ります。
- ソフト・オーバーラン上限がプロセスごとの上限に到達または超えたら、オーバーランはハード・オーバーランとして処理されます。プロセスは次回の**`fbswait`**の呼び出しをブロックし、そのプロセスに対して次回通常どおりスケジュールされたFrequency-Based Schedulerの起動が生じる時、プロセスは**`fbswait`**コールから戻りステータス値が0を返します。ハード・オーバーランの場合はステータスが2を返さないことに注意して下さい。

デッドライン違反の検出

スリープさせる FBS ライブラリ・ルーチンを呼び出す前にスケジュールされたプロセスが適用されたデッドライン時間を超えた場合は*デッドライン違反*が発生します。デッドライン違反が検出された時、適用したデッドラインがデッドライン違反で FBS を停止するように指定している場合はスケジューラーは停止します。

デッドラインは FBS でスケジュールされた各プロセスにオプションで適用することが可能です。更に、各デッドラインは検出に対してスケジューラーを停止、またはスケジューラーを実行したままにするように構成することも可能です。処理の最後でスケジュールされたプロセスがスリープさせる FBS ライブラリ・ルーチンを呼び出す時、デッドラインが検出されます。デッドライン違反の早期検出のため、他の FBS ライブラリ・ルーチンはデッドライン違反に関する明示的なテストを提供します。

デッドライン違反の検出でデッドラインは FBS を停止するように構成することが可能です。これは不安定なアプリケーションをデバッグ、(例えば)外部ハードウェアと結合時の致命的な障害を回避するために利用することが可能です。

FBS デッドラインは*デッドライン型*DEADLINE_CLEAR を使ってプロセスから削除すること、デッドライン型 DEADLINE_WALL_TIME を使って設定することも可能です。DEADLINE_WALL_TIME デッドラインをプロセスに適用する場合に CLOCK_MONOTONIC タイミング・ソースが実行時間を計測するために使用されます。この時、他のクロックの種類はサポートされません。

各デッドライン時間は適切な*デッドライン起点*から計測される必要があります。タイミングの要求に応じて、FBS は 2 つの異なるデッドライン起点をサポートします：

- タスクがメジャー・フレームのマイナー・サイクル内の一定時間で完了する必要がある場合は*サイクル相対デッドライン*が使用されます。例えば、メジャー・フレームあたり100個のマイナー・サイクルを持ち、各マイナー・サイクルが10ミリ秒の持続時間のスケジューラーを構成し、プロセスは周期が10で開始が2サイクル目で実行するようにスケジュールされた場合、100ミリ秒ごとに1回実行(2サイクル目から開始)されます。100ミリ秒ごとに8サイクル目で生成するデータがハードウェアに書き込む準備が出来ている必要がある場合、プロセスは60ミリ秒以内で完了する必要があります。サイクル相対デッドラインを60ミリ秒に設定することで、プロセスがその仕事を終えるのに60ミリ秒より長くかかる場合はプログラムは検出してFBSを停止する可能性があります。
- タスクが実行を開始したら一定の持続時間内にその仕事を完了する必要がある場合は*タスク相対デッドライン*が使用されます。例えば、Process-1とProcess-2が同じサイクルでスケジュールされ、Process-1がより高いリアルタイム優先度である場合、Process-1はProcess-2より前に実行するようにスケジュールされ、Process-2はProcess-1がスリープするとすぐに実行されます。完了までに12ミリ秒を決して超えないことが分かっているProcess-2の実行時間が時折長いのを検出したい場合、Process-2の時間が掛かりすぎる時にFBSを停止するために12ミリ秒のタスク相対デッドラインを使用することが可能です。

加えて、NightSim GUI は時間単位の代わりにサイクルで計測されるデッドラインの他の形態をサポートします。タスクが次のサイクルの開始前、またはリアルタイム・クロックの頻度によって異なるサイクル内の比例ポイント(例えば 75%)の前に完了する必要がある場合は、*サイクル・カウント・デッドライン*を指定することが可能です。詳細については ***NightSim RT User's Guide*** を参照して下さい。

NOTE

デッドラインによるスケジューリングは RedHawk version 5.2.1 以降でサポートされています。

ユーザー・インターフェース

FBS は以下のいずれかを使ってアクセスします：

- リアルタイム・コマンド・プロセッサー**rtcp**。本プログラムはFBS用のコマンド・インタプリターの機能を果たし、キーボードからコマンドを入力する、またはスクリプトを起動することで主要なFBSの操作を実行することが可能です。**rtcp**についてはコマンドを実行する順番を示す図を含めて5章で説明しています。
- 一連のライブラリ・ルーチン。このインターフェスは、CまたはFORTRANで書かれたアプリケーション・プログラムからスケジューラーに関連する機能の全ての範囲を実行することが可能です。Cライブラリ・ルーチンについては6章で説明しています。FORTRANライブラリ・ルーチンについては7章で説明しています。呼び出す可能性のあるルーチンの順番を示す図が含まれています。
- NightSim。これはFBS機能の全ての範囲に対するグラフィカル・ユーザー・インターフェース(GUI)です。NightSimは*NightSim RT User's Guide*で説明しています。

Frequency-Based Schedulerプロセスのデバッグ

Frequency-Based Scheduler でスケジュールされたプロセスのデバッグは、汎用のソースレベル・デバッガ NightView を使って行うことが可能です。

C の実行可能プログラムをデバッグできるようにするには、**-g** オプションを指定してソース・プログラムをコンパイルする必要があります。

FBS でスケジュールされたプロセスをデバッグするために使用可能な NightView のコマンドは次のとおりです：

attach	実行中のプロセスに接続します。本コマンドは既に実行中のプロセスをデバッグすることが可能となります。
detach	接続したプロセスから分離します。本コマンドはデバッガの制御から接続したプロセスを解放することが可能となります。

FBS でスケジュールされたプロセスをデバッグするために NightView を使用するには、プロセス ID (PID)を提供する必要があります。**ps(1)** コマンドを使用することで FBS でスケジュールされたプロセスの PID を簡単に取得することが可能です。C ライブラリ・ルーチン **nametopid(3)** を使用することで選択したプロセス名称の PID を取得することも可能です。これらの各ルーチンの使用については対応する man ページで説明しています。

NightView が実行中のプロセスに接続するには、デバッガの実効ユーザーとグループ ID がデバッガで制御するプロセスの実効ユーザーとグループ ID が一致している必要があります。

attach と **detach** コマンドを使用するための手順に関する追加情報については、*NightView RT User's Guide* を参照して下さい。

3章

タイミング・ソース

概要 3-1
リアルタイム・クロックの使用 3-1
　リアルタイム・クロック・デバイスの理解 3-1
　ユーザー・インターフェースの理解 3-2
　リアルタイム・クロックの使用に関する基本手順 3-2
エッジ・トリガー割込みの使用 3-3
　エッジ・トリガー割込みの理解 3-3
　ユーザー・インターフェースの理解 3-4
ユーザー提供タイミング・デバイスの使用 3-4

3章

タイミング・ソース

本章は FBS スケジューリングの様々なタイミング・ソースの利用に関する手順を含みます。

概要

Real-Time Clock and Interrupt Module (RCIM)は Frequency-Based Scheduler のためのタイミング・ソースとして適したリアルタイム・クロックとエッジ・トリガー割込みを提供します。各 RCIM は最大 8 個のリアルタイム・クロック・タイマー(RTC)と最大 12 個のエッジ・トリガー割込み(ETI)を提供します。これらはローカルで使用または RCIM チェーンに接続されたシステム間で分配することが可能です。ユーザー定義デバイスもタイミング・ソースとして使用することが可能です。

スケジューラーを実行する前にタイミング・ソースはスケジューラーに「接続」することで選択されます。リアルタイム・クロックを使用する場合、カウントと分解能の値を設定することでマイナー・サイクルの持続時間を定義することが可能です。

続くセクションで利用可能なタイミング・ソースに関する情報を提供します。

リアルタイム・クロックの使用

本セクションは Frequency-Based Scheduler のタイミング・ソースとして RCIM リアルタイム・クロックを使用するための情報を提供します。論考は次を含みます：

- リアルタイム・クロックの概要
- デバイスへのユーザー・インターフェースの説明
- デバイスを使用するための基本手順

リアルタイム・クロック・デバイスの理解

リアルタイム・クロック(RTC)デバイスは様々なタイミングや頻度の制御機能用に設計されています。これはある範囲のクロック・カウント値と分解能一式を基にして作り出すいくつかの異なるタイミング間隔を提供します。これはとりわけ FBS スケジューリングに相応しい機能になります。

Real-Time Clock and Interrupt Module (RCIM)は各システムで最大 8 個のリアルタイム・クロック(0-7)を提供します。複数のシステムが RCIM チェーンを使って接続されている場合、全ての RTC は分配するように指定することが可能、つまり、割り込みは接続された全てのシステムに送信されます。分配された RTC は RCIM チェーン内のどのシステムでも配置することが可能です。

各 RTC はそれ自身のキャラクター特殊デバイス・ファイルを通して参照されます：

```
/dev/rcim/rtcN
```

詳細については、**rcim(4)** の man ページまたは *Real-Time Clock and Interrupt Module (RCIM) User's Guide* を参照して下さい。

ユーザー・インターフェースの理解

Frequency-Based Scheduler のタイミング・ソースとしてリアルタイム・クロックを使用している場合、FBS の任意のインターフェースはスケジューラーにクロックを接続またはスケジューラーからクロックを分離、もしくはクロックを設定、開始、停止するために使用することが可能です。実行される動作および使用される **rtcp** コマンドや C および FORTRAN ライブラリ・ルーチンは後述の「リアルタイム・クロックの使用に関する基本手順」項で説明します。FBS でスケジュールされたリアルタイム・クロックの選択および操作するための NightSim の使用は *NIghtSim RT User's Guide* で説明します。

リアルタイム・クロックへのインターフェースとして **rtcp** または C ライブラリに含まれているルーチンを使用することを推奨します。**rtcp** コマンドは殆どの初期化タスクから分離しているので最も簡単に使用できます。

リアルタイム・クロックはシステムコール **open(2)**、**close(2)**、**ioctl(2)** を使うことで直接制御することが可能です。デバイスは **read(2)** と **write(2)** のシステムコールをサポートしないことに注意して下さい。詳細は **rcim_rtc(4)** の man ページを参照して下さい。

リアルタイム・クロックの使用に関する基本手順

リアルタイム・クロックへのインターフェースとして使用するために選んだものが **rtcp** であろうがライブラリに含まれているルーチンであろうが、FBS でスケジュールされたリアルタイム・クロックの使用に関する基本手順は同じです：

手順1： Frequency-Based Schedulerのリアルタイム・クロックに接続

手順2： クロック・カウントの値とクロック・カウントあたりの分解能を指定してマイナー・サイクルの持続時間を設定

手順3： リアルタイム・クロックのカウントを開始

手順4： シミュレーションを開始

手順5： シミュレーションの停止

手順6： リアルタイム・クロックのカウントを停止

手順7： リアルタイム・クロックを分離

各手順に対応する **rtcp** コマンドおよび C と FORTRAN のルーチンを表 3-1 に示します。

表3-1 RTCを使用するためのrtcpコマンドとルーチン

手順	rtcp コマンド	C ルーチン	FORTRAN ルーチン
1	ats	fbsattach	fbsattach
2	stc	fbssetrtc	fbssetrtc
3	rc	fbsrunrtc	fbsrunrtc
4	start	fbsintrpt	fbsintrpt
5	stop	fbsintrpt	fbsintrpt
6	sc	fbsrunrtc	fbsrunrtc
7	dts	fbsdetach	fbsdetach

rtcp コマンドの説明については 5 章、C ライブラリに含まれているルーチンの説明については 6 章、FORTRAN ルーチンについては 7 章を参照して下さい。

エッジ・トリガー割込みの使用

本セクションは Frequency-Based Scheduler のタイミング・ソースとしてエッジ・トリガー割込みを使用するための情報を提供します。論考は次を含みます：

- エッジ・トリガー割込みの概要
- デバイスへのユーザー・インターフェースの説明

エッジ・トリガー割込みの理解

エッジ・トリガー割込み(ETI)デバイスは RCIM により提供される外部割込みラインへのソフトウェア・インターフェースです。外部割込みラインは、信号パルスを生成する任意のユーザー・デバイスからコンピュータ・システムに入ってくる外部割込みをシステムが検出するための手段を提供します。この信号パルスは Frequency-Based Scheduler のタイミング・ソースとして使用することが可能です。

RCIM あたり最大 12 個のエッジ・トリガー割込みがあります。複数のシステムが RCIM チェーンを介して接続されている場合、全ての ETI は分配するように指定することが可能、つまり、割り込みは接続された全てのシステムに送信されます。分配された ETI は RCIM チェーン内のどのシステムでも配置することが可能です。

各 ETI はそれ自身のキャラクター特殊デバイス・ファイルを通して参照されます：

```
/dev/rcim/etiN
```

詳細については、**rcim(4)** の man ページまたは *Real-Time Clock and Interrupt Module (RCIM) User's Guide* を参照して下さい。

ユーザー・インターフェースの理解

Frequency-Based Scheduler のタイミング・ソースとしてエッジ・トリガー割込みを使用している場合、FBS の任意のインターフェースはスケジューラーに割り込みを接続またはスケジューラーから割り込みを分離するために使用することが可能です。シミュレーションを開始する時に既に割り込みが生成されていることを確かめて下さい。

割り込みとスケジューラを接続および分離するために使用される **rtcp** コマンドおよび C と FORTRAN ライブラリ・ルーチンを表 3-2 に示します。

表3-2 ETIを使用するためのrtcpコマンドとルーチン

rtcp コマンド	C ルーチン	FORTRAN ルーチン
ats	fbsattach	fbsattach
dts	fbsdetach	fbsdetach

rtcp コマンドの説明については 5 章、C ライブラリに含まれているルーチンの説明については 6 章、FORTRAN ルーチンについては 7 章を参照して下さい。FBS でスケジュールされたリアルタイム・クロックの選択および操作するための NightSim の使用は *NIghtSim RT User's Guide* で説明します。

エッジ・トリガー割込みはシステムコール **open(2)**、**close(2)**、**ioctl(2)**を使うことで直接制御することが可能です。デバイスは **read(2)**、**write(2)**、**mmap(2)**のシステムコールをサポートしないことに注意して下さい。詳細は **rcim_eti(4)** の man ページを参照して下さい。

ユーザー提供タイミング・デバイスの使用

Frequency-Based Scheduler のタイミング・ソースとしてお手持ちのデバイスを使用することを望むかもしれません。お手持ちのデバイスを使用するには次を確認する必要があります：

- デバイス・ドライバーが**ioctl**コールのIOCTLVECNUMをサポートしている
- デバイス・ドライバーが**ioctl**コールのIOCTLKEEPALIVEをサポートしている
- デバイスが一連の割り込みを生成する

タイミング・ソース・ドライバーにFBS割り込みハンドラ・ルーチンに関連付けられたIRQを提供するため、FBSは**ioctl**のIOCTLVECNUMを呼び出します。タイミング・ソース・ドライバーは割り込みハンドラからinvoke_irq()を呼び出してその割り込みハンドラを呼び出す必要があります。

コールは以下の指定が必要です：

```
#include <linux/ioctl/vecnum.h>

ioctl (fd, IOCTLVECNUM, arg);
```

fd　　デバイスのファイル記述子

IOCTLVECNUM　タイミング・ソース・ドライバーにIRQを渡すためのコマンド

arg　　本ドライバーに必然的に接続される割り込みハンドラに加えて、呼び出される割り込みハンドラのIRQ。値ゼロ(0)は以前の関連性を無効にします。

デバイスを最終的にクローズさせる時に関連するデバイスをシャット・ダウンさせないようにするには、以下も提供します：

```
#include <linux/ioctl/keepalive.h>

ioctl (fd, IOCTLKEEPALIVE, arg);
```

arg　　非ゼロの場合、可能ならばデバイスを最後にクローズした後もデバイスが実行し続けることを可能にするキープアライブを有効にします。ゼロ(0)の場合、キープアライブ状態を無効にします。

割り込みハンドラのボトムで、与えられたIRQのハンドラを呼び出すために以下が行われます：

```
#include <linux/interrupt.h>

if (irq)
    invoke_irq (irq, (struct pt_regs *)0);
```

このインターフェスを実装するLinuxドライバーの実例は以下のRedHawkカーネル・ソース・ディレクトリで見ることが可能です：

```
<kernel-sourcedir>/drivers/char/rcim
```

4章

Performance Monitorの使用

Performance Monitorとは何か? 4-1
　何の値を監視するのか? 4-1
　アイドル時間の監視 4-3
　非スケジュール・プロセスの監視 4-4
ユーザー・インターフェース 4-5
シミュレーション性能の最適化 4-5

4章

Performance Monitorの使用

本章は Performance Monitor とその機能についての説明を含んでいます。監視する値とシステムのアイドル時間および非スケジュール・プロセスの監視を有効にする方法について説明します。シミュレーションの性能を最適化する提案を提供し、ユーザー・インターフェースの順序を紹介します。

Performance Monitorとは何か？

Performance Monitor (PM)は Frequency-Based Scheduler を使ってスケジュールされたプロセスで CPU の使用状況を監視することを可能にします。

Performance Monitor は次の機能を提供します：

- プロセスまたはプロセッサによりPerformance Monitorの値を取得
- 1つのプロセッサーから全てのPerformance Monitorの機能を制御(つまり、任意のプロセッサーに対して性能監視を有効にします)
- プロセスまたはプロセッサによりPerformance Monitorを開始および停止
- プロセスまたはプロセッサによりPerformance Monitorの値を消去
- 割り込み時間を含めるまたは除外するためのタイミング・モードを設定

Performance Monitor を使用するために高分解能プロセス・アカウンティングをシステムに構成する必要があることに注意して下さい。詳細については 1 章の「構成」項を参照して下さい。

何の値を監視するのか？

Performance Monitor は、Frequency-Based Scheduler により起動された時間から **`fbswait(3)`** が呼ばれるまでにプロセスが実行に費やした時間の経過を追います。時間はマイクロ秒で測定されます。

スケジューラーにより起動されるプロセスのある段階を*反復*または*サイクル*と呼びます。FBS でスケジュールされたプロセスの PM の値はサイクルまたは反復の単位、および*メジャー・フレーム*の単位の両方が報告されます。これらは最後に Performance Monitor の値が消去されて Performance Monitor が有効になった後に発生したことを反映します。

割り込み時間を含めるために Performance Monitor タイミング・モードが設定された場合、プロセスのユーザーおよびシステム時間はプロセスが実行中に生じる経過時間を合計します(割り込みを処理するために費やした時間はプロセスのシステム時間に加算されます)。割り込み時間を除外するために Performance Monitor タイミング・モードが設定された場合、プロセスのユーザーおよびシステム時間はプロセスが現在処理を実行中である時に生じる時間を合計し、割り込みの処理に費やした時間は除外します。設定されたタイミング・モードに割り込み時間を含めようが除外しようが、コンテキスト・スイッチは常に新しいプロセスのシステム時間に含まれます。

有効になった Performance Monitor が FBS でスケジュールされたプロセスが 1 つであろうが FBS でスケジュールされたプロセスがプロセッサ上の全てであろうが、以下の値のタイプが各プロセスに対して維持されます：

合計反復/サイクル数
: プロセスがスケジューラーにより起動された回数

最新時間
: プロセスが最後にスケジューラーにより起動されてから**fbswait(3)**が呼ばれるまでの実行に費やされた時間

合計時間
: プロセスが全てのサイクルで実行に費やした時間の合計

最小サイクル時間
: プロセスがサイクルで実行に費やした時間の最小

最小サイクル・サイクル
: 最小サイクル時間が発生したマイナー・サイクルの番号

最小サイクル・フレーム
: 最小サイクル時間が発生したメジャー・フレームの番号

最大サイクル時間
: プロセスがサイクルで実行に費やした時間の最大

最大サイクル・サイクル
: 最大サイクル時間が発生したマイナー・サイクルの番号

最大サイクル・フレーム
: 最大サイクル時間が発生したメジャー・フレームの番号

最小フレーム時間
: プロセスがメジャー・フレーム中に実行に費やした時間の最小

最小フレーム・フレーム
: 最小フレーム時間が発生したメジャー・フレームの番号

最大フレーム時間
: プロセスがメジャー・フレーム中に実行に費やした時間の最大

最大フレーム・フレーム
: 最大フレーム時間が発生したメジャー・フレームの番号

アイドル時間の監視

Performance Monitor はプロセスの*アイドル時間*を監視することが可能です。アイドル時間は CPU がビジーではない時間を指します。

プロセスのアイドル時間を監視することで、更なるプロセスへの割り当てに利用可能な CPU 時間の総量を測定することが可能です。

/idle プロセスを Frequency-Based Scheduler に追加して希望するプロセッサ上にスケジューリングすることでプロセッサーのアイドル時間を監視します。複数回選択した Frequency-Based Scheduler に**/idle** を追加して異なるプロセッサ上に毎回スケジューリングすることで複数のプロセッサに対してアイドル時間を監視することが可能です。**/idle** を複数の Frequency-Based Scheduler に追加することも可能です。一方、特定のプロセッサーに一度だけ**/idle** をスケジュールすることが可能であるという事に注意することが重要となります。

Frequency-Based Scheduler に**/idle** を追加するには、以下のいずれかを行ってください：

- **rtcp**の**sp**コマンドを実行
- Cプログラムから**sched_pgmadd(3)**呼び出しを行う
- FORTRANプログラムから**schedpgmadd(3f)**呼び出しを行う

sp コマンドの説明は 5-32 ページで提供します。**sched_pgmadd(3)** ルーチンの説明は 6-51 ページおよび man ページで見ることが可能です。**schedpgmadd(3f)** ルーチンの説明は 7-55 ページおよび man ページで見ることが可能です。

Frequency-Based Scheduler に**/idle** を追加する場合、指定する必要のある唯一のパラメータは CPU です。**/idle** のデフォルトのスケジューリング優先度はゼロです。開始ベース・サイクルはゼロで、周期は 1 となります。**/idle** プロセスはマイナー・サイクルごとにスケジュールされ、各メジャー・フレームの最初のマイナー・サイクルから開始します。

NOTE

スケジューラーに**/idle** を追加するため **sched_pgmadd(3)** サブルーチンを使用する場合、プロセッサを指定するビット・マスクに設定可能なのは 1 ビットだけとなります。スケジューラーに**/idle** を追加して複数のプロセッサにスケジュールするには、サブルーチンを繰り返し呼び出して各呼び出しで異なるプロセッサーを指定する必要があります。

/idle がスケジュールされた後、一意の Frequency-Based Scheduler 識別子が返されます。FBS または PM に関連するタスクを実行する場合は、**/idle** を識別するこの数字を使用して下さい。

以下のいずれかを行うことで、FBS でスケジュールされた他のプロセスについても**/idle** を取得したのと同じ方法でスケジューリング情報を取得します：

- **rtcp**の**vp**コマンドを実行
- Cプログラムから**sched_fbsqry(3)**または**sched_pgmqry(3)**呼び出しを行う
- FORTRANプログラムから**schedfbsqry(3f)**または**schedpgmqry(3f)**呼び出しを行う

vp コマンドの説明は 5-35 ページで提供します。**sched_fbsqry(3)** ルーチンの説明は 6-36 ページ、**sched_pgmqry(3)** は 6-61 ページ、**schedfbsqry(3f)** は 7-52 ページ、**schedpgmqry(3f)** は 7-59 ページで提供され、各々の man ページに説明があります。

/idle がスケジュールされているプロセッサまたはプロセス自体の Performance Monitor を有効にした場合、全ての Performance Monitor の値を取得することが可能です。

非スケジュール・プロセスの監視

Performance Monitor は*非スケジュール・プロセス*の性能を監視する追加機能を提供します。非スケジュール・プロセスとはスケジューラーによって起動されず、**fbswait** を呼び出さないものです。

そのようなプロセスの PM の値を取得できるようにするには、それらを Frequency-Based Scheduler に使いし、開始ベース・サイクルをゼロそして周期をゼロに指定します。指定する必要のある他のスケジューリング・パラメータはプロセス・スケジューリング優先度とそれをスケジュールする CPU が含まれます。オプションで Frequency-Based Scheduler にスケジュールされたプロセスに渡す 8 進数の値を指定します。オーバーラン停止フラグは非スケジュール・プロセスには適用されません。

以下のいずれかを行うことで、Frequency-Based Scheduler に非スケジュール・プロセスを追加します：

- **rtcp**の**sp**コマンドを実行
- Cプログラムから**sched_pgmadd(3)**呼び出しを行う
- FORTRANプログラムから**schedpgmadd(3f)**呼び出しを行う

sp コマンドの説明は 5-32 ページで提供します。**sched_pgmadd(3)** ルーチンの説明は 6-51 ページ、**schedpgmadd(3f)** ルーチンの説明は 7-55 ページで提供され、各々の man ページに説明があります。

プロセスがスケジュールされた後、一意の Frequency-Based Scheduler 識別子が返されます。それ以降、FBS または PM に関連するタスクを実行する場合は、プロセスを識別するこの数字を使用することが可能です。

以下のいずれかを行って他の FBS でスケジュールされたプロセスを取得するのと同じ方法で非スケジュール・プロセスに関するスケジューリング情報を取得します：

- **rtcp**の**vp**コマンドを実行
- Cプログラムから**sched_pgmqry(3)**呼び出しを行う
- FORTRANプログラムから**schedpgmqry(3f)**呼び出しを行う

vp コマンドの説明は 5-35 ページで提供します。**sched_pgmqry(3)** ルーチンの説明は 6-61 ページ、**schedpgmqry(3f)** は 7-59 ページで提供され、各々の man ページに説明があります。

非スケジュール・プロセスに対して保持される PM の値は以下を含みます：

- 最新時間
- 合計時間
- 最小フレーム時間および発生したフレームの番号
- 最大フレーム時間および発生したフレームの番号

プロセスがスケジュールされたプロセッサーまたは個々のプロセスへの Performance Monitor を有効にすることで上記の値を取得します。

ユーザー・インターフェース

PM は以下のいずれを使ってアクセスします：

- リアルタイム・コマンド・プロセッサー**`rtcp`**。本プログラムはPerformance Monitor用のコマンド・インタプリターの機能を果たし、キーボードからコマンドを入力する、またはスクリプトを起動することで主要なPMの実行の操作を実行することが可能です。**`rtcp`**についてはコマンドを実行する順番を示す図を含めて5章で説明しています。
- 一連のライブラリ・ルーチン。このインターフェスは、CおよびFORTRANで書かれたアプリケーション・プログラムからPerformance Monitorに関連する機能の全ての範囲を実行することが可能です。ライブラリ・ルーチンについては6章および7章で説明しており、各々、呼び出す可能性のあるルーチンの順番を示す図が含まれています。
- NightSim。これはPM機能の全ての範囲に対するグラフィカル・ユーザー・インターフェース(GUI)です。NightSimは*NightSim RT User's Guide*で説明しています。

シミュレーション性能の最適化

リアルタイム処理でマルチプロセッサー・システムを使用する利点の 1 つは、複数のプロセッサー間でプロセスを分散することでシミュレーションの性能を最適化することが可能であることです。

シミュレーションを構成するプログラムをスケジュールするために FBS を使用すると、プロセスが CPU を使用している範囲を確認し、指定した頻度で実行しているかどうかを調べるために Performance Monitor を使用することが可能となります。

プログラムは Frequency-Based Scheduler に追加される時にプロセッサーにスケジュールされます。スケジュールされるプロセッサーはスケジューラーに追加される時に指定された CPU バイアス(アフィニティ)により決定されます。プログラムがスケジュールされた後、複数のプロセスまたはプロセッサーで Performance Monitor が有効になりシミュレーションが実行されます。

FBS でスケジュールされた各プロセスについて維持される PM の値を調査することで、次を確認することが可能です：

- プロセスが割り当てられたプロセッサー
- プロセスが実行に費やした時間
- 割り当てた頻度で実行していないプロセス

プロセスが割り当てた頻度で実行されていないことが分かった場合、頻度、他のプロセスに使用されている CPU 時間、全プロセスの CPU バイアスを調査して下さい。プロセスの実行が長すぎる場合、事実を正確にとらえるためにデッドラインを使用することが可能です。

プロセスの CPU バイアスが複数のプロセッサであることを識別した場合、動的負荷バランスとなるためにプロセスがビット・マスクで指定した特定の CPU 上で費やした時間がどれくらいであるかを判定することは出来ないことに注意して下さい。動的負荷バランスを避けるには、ビット・マスクで 1 つのプロセッサーだけを指定して下さい。Performance Monitor を使用することで、プロセスが割り当てられた CPU 上で費やした時間がどれくらいかを知ることが可能となります。必要に応じてプロセスを再分散することが可能です。

アイドル時間の Performance Monitor についても有効にすることが可能です。そのための手順は本章の「アイドル時間の監視」で説明しています。各プロセッサーのアイドル時間を調査することで、最も負荷の軽いプロセッサーを特定することやリアルタイム・プロセスをスケジューリングするために使用可能な更なる CPU 時間を計算することが可能となります。

FBS でスケジュールされたプロセスや選択したプロセッサーのアイドル時間に関する PM の値を解析することでシミュレーションに最適なプロセッサー割り当てを決定することが可能です。必要に応じて CPU バイアスを変更して FBS でスケジュールされたプロセスを再分散して下さい。

そのようにするには、最初にスケジュールされたプロセスをスケジューラーから削除してスケジューラーにプロセスを追加する必要があることに注意することが重要です。Frequency-Based Scheduler の概要およびその用途に対応するインターフェースについては、2 章を参照して下さい。

タスクの性能はどのようにシステムの CPU が構成されているかによって影響を与える可能性があります。CPU シールディングやハイパースレッドの詳細については *RedHawk Linux User's Guide* の「リアルタイム性能」章を参照して下さい。

5章

rtcpの使用

リアルタイム・コマンド・プロセッサーとは何か? 5-1
rtcpとFrequency-Based Scheduler 5-1
rtcpとPerformance Monitor 5-2
実行モード 5-2
ダイレクト・モードの使用 5-2
コマンド名と引数を使ったrtcpの起動 5-3
スクリプト・ファイルからリダイレクトされたコマンドを使ったrtcpの起動. . 5-3
rtcpを呼び出すスクリプトの起動 5-3
対話モードの使用 5-4
困ったときは 5-5
rtcpコマンド 5-7
rtcpコマンドの要約 5-7
コマンドの順序 5-8
rtcpコマンドの使用 5-10
ats – Frequency-Based Schedulerにタイミング・ソースを接続 5-11
chs – Frequency-Based Schedulerのアクセス権を変更 5-12
cs – Frequency-Based Schedulerを構成 5-13
dts – Frequency-Based Schedulerからタイミング・ソースを分離 5-15
rms – Frequency-Based Schedulerを削除 5-16
svs – スケジューラの構成を保存 5-17
vc – マイナー・サイクル/メジャー・フレームの総数を表示 5-18
vs – スケジューラーの構成を表示 5-19
ls – システム上の全スケジューラーを表示 5-21
rc – リアルタイム・クロックを開始 5-21
sc – リアルタイム・クロックを停止 5-22
stc – リアルタイム・クロックを設定 5-22
gtc – リアルタイム・クロックの設定を表示 5-23
start – Frequency-Based Schedulerのスケジューリングを開始 5-24
resume – Frequency-Based Schedulerのスケジューリングを再開 5-25
stop – Frequency-Based Schedulerのスケジューリングを停止. 5-25
rmp – Frequency-Based Schedulerからプロセスを削除 5-26
rsp – プロセスを再スケジュール 5-28
sp – Frequency-Based Schedulerのプロセスをスケジュール 5-32
vp – Frequency-Based Schedulerのプロセスを表示 5-35
cpm – Performance Monitorの値を消去 5-38
pm – Performance Monitorを開始/停止 5-40
vcm – Performance Monitorのタイミング・モードを表示/修正 5-42
vpm – Performance Monitorの値を表示 5-43
ex – リアルタイム・コマンド・プロセッサーを終了 5-47
he – ヘルプ情報を表示 5-47

5章

rtcpの使用

本章ではリアルタイム・コマンド・プロセッサー**rtcp** について説明します。これは実行モード、FBS や Performance Monitor でスケジュールするために使用するコマンドの順序、ヘルプの使用法、各コマンドの詳細について説明しています。

リアルタイム・コマンド・プロセッサーとは何か？

リアルタイム・コマンド・プロセッサー(**rtcp**)は Frequency-Based Scheduler や Performance Monitor のコマンド・インタプリターの機能を果たすプログラムです。**rtcp** はキーボードからコマンドを入力する、またはシェル・コマンド・ラインからスクリプトを起動することで主要な操作を実行することが可能です。これは関連するサービスを実行するための要求としてコマンドを読み取りそれらを解釈します。

実行のモードはダイレクトと対話の 2 つが存在します。これらのモードは 5-2 ページで詳細を説明しています。

rtcp はコマンドや引数に関するオンライン情報を取得できるようにするヘルプ機能も備えています。ヘルプ機能を利用するための手順については 5-5 頁で説明しています。

rtcpとFrequency-Based Scheduler

FBS に関連する **rtcp** コマンドは以下のような主要な操作の実行を有効にします：

- スケジューラーの構成
- プログラムのスケジューリング
- スケジューラー構成の保存
- タイミング・ソースの設定
- シミュレーションの実行
- ステータスの問合せ

Frequency-Based Scheduler に関連する **rtcp** コマンドの要約については表 5-1 を参照して下さい。Frequency-Based Scheduler の概要は 2 章で提供しています。**rtcp** コマンドを使用する前にその章と Performance Monitor に関する章を読むことを推奨します。

rtcpとPerformance Monitor

Performance Monitor に関連する **rtcp** コマンドは以下のような主要な操作の実行を有効にします：

- Performance Monitorの値の消去
- Performance Monitorの開始および停止
- スケジューラー構成の保存
- タイミング・モードの設定
- 値の問合せ

Performance Monitor に関連する **rtcp** コマンドの要約については表 5-1 を参照して下さい。Performance Monitor の概要は 4 章で提供しています。**rtcp** コマンドを使用する前にその章と Frequency-Based Scheduler に関する章を読むことを推奨します。

実行モード

リアルタイム・コマンド・プロセッサーはコマンドの実行に 2 つのモードを提供します：

ダイレクト・モード	シェル・コマンド・ラインからリアルタイム・コマンド・プロセッサーのコマンドを起動します
対話モード	シェル・コマンド・ラインからリアルタイム・コマンド・プロセッサー自体を起動した後、コマンド・プロセッサー内から目的のコマンドを入力できます

ダイレクト・モードの使用

以下のいずれかの方法でダイレクト・モードを使用して下さい。各手法については後述します。

- システム・コマンド・プロンプトでコマンド名とその引数と共にリアルタイム・コマンド・プロセッサーを起動。
- システム・コマンド・プロンプトでリアルタイム・コマンド・プロセッサーを起動して、ターミナル・キーボードの代わりにファイルからの標準入力をリダイレクト。
- システム・コマンド・プロンプトでスクリプトを起動。

コマンド名と引数を使ったrtcpの起動

コマンドと引数を使って **rtcp** を呼び出す場合は以下の書式を使用して下さい：

```
$ rtcp command [-option [argument]] [-option [argument]] ...
```

一度にコマンド・ラインで入力できるのは 1 つのコマンドだけであることに注意して下さい。コマンドおよびその引数を入力するのに複数行が必要な場合、次の行に進むために行の最後にバックスラッシュ(\)または円記号(¥)を入力して下さい。

スクリプト・ファイルからリダイレクトされたコマンドを使ったrtcpの起動

この方法は実行したい **rtcp** コマンドを含むファイルを生成して、そのファイルを **rtcp** への標準入力として使用する必要があります。自由に選んだテキスト・エディタを使って、または **svs** (Save Scheduler Configuration)コマンドを実行してファイルを生成して下さい。**svs** コマンドは 5-17 ページで説明しています。

ファイルを生成するのにテキスト・エディタを使用する場合、各コマンドは別々の行に入力して下さい。以下のいずれかの書式を使用することが可能です：

rtcp *command* [*-option* [*argument*]] [*-option* [*argument*]] ...

または

command [*-option* [*argument*]] [*-option* [*argument*]] ...

コマンドおよびその引数を入力するのに複数行が必要な場合、次の行に進むために行の最後にバックスラッシュ(\)または円記号(¥)を入力して下さい。

ファイルを生成した後、リアルタイム・コマンド・プロセッサーを起動してファイルからの標準入力をリダイレクトして下さい：

```
$ rtcp < rtcp_input_file
```

rtcpを呼び出すスクリプトの起動

この方法は実行したい **rtcp** コマンドを含む実行ファイルを生成する必要があります。ファイルの最初の行はコマンドまたは以下のテキストを含めることが可能です：

#!*program_name*

program_name は起動するシェルを含んでいるファイルの名称を指定します(例えば、**/bin/bash**)。

各コマンドは以下の書式に準じて別々の行に入力する必要があります：

rtcp *command* [*-option* [*argument*]] [*-option* [*argument*]] ...

コマンドおよびその引数を入力するのに複数行が必要な場合、次の行に進むために行の最後にバックスラッシュ(\)または円記号(¥)を入力して下さい。

ファイルを生成した後、**chmod(1)** コマンドを使って実行可能ファイルにしてから他のコマンドのようにコマンド・ラインからそれを起動して下さい：

```
$ chmod 755 rtcp_script
$ rtcp_script
```

chmod (1)コマンドの使用に関する情報については、対応するmanページを参照して下さい。リアルタイム・コマンド・プロセッサー・スクリプトの実例については付録 A を参照して下さい。

対話モードの使用

リアルタイム・コマンド・プロセッサーを対話形式で起動するには以下の手順を利用して下さい：

1. システム・コマンド・プロンプトで **rtcp** コマンドを指定して下さい：

   ```
   $ rtcp
   ```

   リアルタイム・コマンド・プロセッサーのプロンプトが表示されます：

   ```
   rtcp>
   ```

2. プロンプトにて以下の書式を使いリアルタイム・プロセッサーのコマンドを入力して下さい：

   rtcp>*command* [*-option* [*argument*]] [*-option* [*argument*]] ...

   コマンドおよびその引数を入力するのに複数行が必要な場合、その行を継続させるためにバックスラッシュ(\)または円記号(¥)を入力して下さい。

   大抵の場合、コマンドが正常に実行されたらメッセージが表示され、そこには当てはまる構成データ、スケジューリング情報、Performance Monitor の値が表示されます。

   コマンドに関連するメッセージやデータは 5-7 ページから始まる参照情報に含まれています。エラーが発生した場合、エラーの種類を示すメッセージが表示されます。エラー・メッセージは付録 B で一覧表示および説明しています。

3. コマンド・プロセッサーを終了してシェルに戻るには、以下を入力して下さい：

   ```
   rtcp>ex
   ```

   システム・コマンド・プロンプトが再度表示されます。

困ったときは

rtcp のオンラインに関する情報へのアクセス方法は以下が利用可能です：

- **rtcp(1)**のmanページ
- **rtcp**のヘルプ機能

he コマンドを使って **rtcp** のヘルプ機能にアクセスして下さい。提供される情報は以下を含みます：

- **rtcp(1)**の全プロセッサー・コマンドの一覧と簡単な説明
- 特定コマンドの説明と書式
- 全コマンドの引数の一覧と説明

全コマンドの一覧を表示するには、以下のように **he** コマンドを入力して下さい：

```
he
```

コマンドが画面 5-1 で示すように表示されます。

画面5-1 rtcpコマンドの表示

```
          rtcp commands

ats - attach timing source to FBS              chs - modify FBS access permissions
cs  - configure FBS                            dts - detach timing source from FBS
rms - remove FBS                               svs - save FBS configuration to a file
vc  - view current frame/cycle count           vs  - view FBS configuration
ls  - list FBSs

rc  - run real-time clock                      sc - stop real-time clock
stc - set real-time clock values               gtc- get real-time clock values

start - start FBS                              resume - resume FBS
stop  - stop FBS

rmp - remove a process on a FBS                rsp- reschedule a process on a FBS
sp  - schedule a process on a FBS              vp - view scheduled processes on FBS

cpm - clear performance monitor tables         pm - start/stop performance monitor
vcm - view/modify PM timing mode               vpm- view performance

he - help                                      ex - exit rtcp
```

特定コマンドの説明を表示するには、引数としてそのコマンド名を添えて **he** を入力して下さい。例えば、**ats** コマンドの説明を表示するには、次のように指定します：

```
he ats
```

ats コマンドの説明およびコマンドを入力するための書式は以下のように表示されます：

```
Attach timing source to a FBS
rtcp ats -s scheduler -d device | -e
```

コマンドの引数の一覧を表示するには、引数として **option** を添えて **he** を入力して下さい：

```
he option
```

rtcp 引数の初期画面は画面 5-2 に示されるように表示されます。

画面5-2 rtcp引数の初期画面

```
          rtcp parameters

-a                  remove program from FBS and terminate
-b {F|R|O}          scheduling policy
-c cpu_bias         cpu bias (* = all CPUs) (default = current CPU)
-d name             devicename or filename
-e                  EOC flag
-f frequency        number of minor cycles to next wakeup (default = 1)
-h {halt|nohalt}    halt FBS on deadline violation (default = nohalt)
-i fpid             process fpid number (default = -1)
-m start_cycle      1st minor cycle to wakeup (default = 0)
-n proc_name        process name
-o {halt|nohalt}    halt FBS on overrun flag (default = nohalt)
-p priority         process priority
-r {cycle|task}     deadline origin flag (default = cycle)
-s scheduler        FBS scheduler key
-t {in|ex}          include or exclude interrupt time in pm monitor
-v parameter        process initiation parameter
-x {av|mi|ma|al}    performance monitor display option (default = average)

Enter 'he op2' for more parameters
```

rtcp 引数の 2 番目の画面を表示するには、引数として **op2** を添えて **he** を入力して下さい：

```
he op2
```

rtcp 引数の 2 番目の画面は画面 5-3 に示されるように表示されます。

画面5-3 rtcp引数の2番目の画面

```
          rtcp parameters

-C cycles/frame      number of minor cycles per major frame
-D duration          clock tick duration (default = 10us)
-G gid               effective group ID for FBS (default = current user)
-I permissions       permissions for FBS in octal (default = 0600)
-L soft_limit        soft overrun limit (default = 0)
-M progs/cycle       maximum number of processes per minor cycle
-N progs/fbs         maximum number of processes per FBS
-O clock_ticks       number of clock ticks per minor cycle
-P {ON|OFF}          enable/disable performance monitor (default = OFF)
-R {-1 | 0 | 1}      reset process flag (default = 0)
-S delay             time to delay, in seconds
-T deadline          deadline microseconds (default = Clear)
-U uid               effective user ID for FBS (default = current user)
```

rtcpコマンド

rtcpコマンドの要約

rtcp コマンドを表 5-1 に示します。各コマンドに関する全ての情報と同様にコマンドの順序のフローチャートを以下に示します。

表5-1 rtcpコマンド

コマンド	ページ	説明
Frequency-Based Scheduler:		
ats	5-11	Frequency-Based Schedulerにタイミング・ソースを接続
chs	5-12	Frequency-Based Schedulerのアクセス権を変更
cs	5-13	Frequency-Based Schedulerを構成
dts	5-15	Frequency-Based Schedulerからタイミング・ソースを分離
gtc	5-23	リアルタイム・クロックの値を表示
ls	5-21	システム上の全Frequency-Based Schedulerの構成を表示
rc	5-21	リアルタイム・クロックの開始

(次ページに続く)

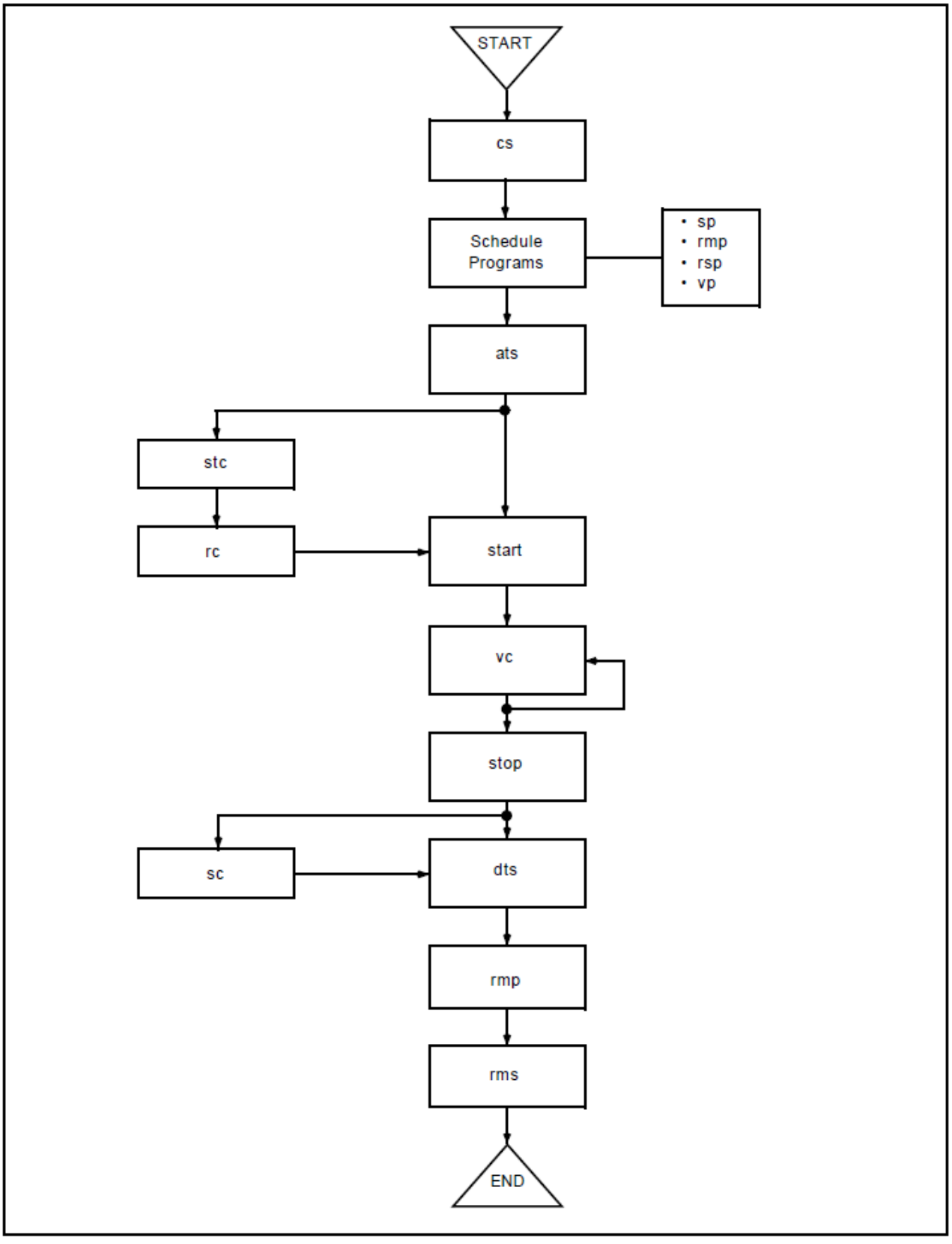
表5-1 rtcpコマンド (続き)

コマンド	ページ	説明
Frequency-Based Scheduler:		
resume	5-25	Frequency-Based Schedulerのスケジューリングを再開
rmp	5-26	Frequency-Based Schedulerからプロセスを削除
rms	5-16	Frequency-Based Schedulerを削除
rsp	5-28	プロセスを再スケジュール
sc	5-22	リアルタイム・クロックを停止
sp	5-32	Frequency-Based Schedulerのプロセスをスケジュール
start	5-24	Frequency-Based Schedulerのスケジューリングを開始
stc	5-22	リアルタイム・クロックの値を設定
stop	5-25	Frequency-Based Schedulerのスケジューリングを停止
svs	5-17	スケジューラの構成を保存
vc	5-18	マイナー・サイクル/メジャー・フレームの総数を表示
vp	5-35	Frequency-Based Schedulerのプロセスを表示
vs	5-19	スケジューラーの構成を表示
Performance Monitor:		
cpm	5-38	Performance Monitorの値を消去
pm	5-40	Performance Monitorを開始/停止
vcm	5-42	Performance Monitorのタイミング・モードを表示/修正
vpm	5-43	Performance Monitorの値を表示
rtcp:		
ex	5-47	リアルタイム・コマンド・プロセッサーを終了
he	5-47	ヘルプ情報を表示

コマンドの順序

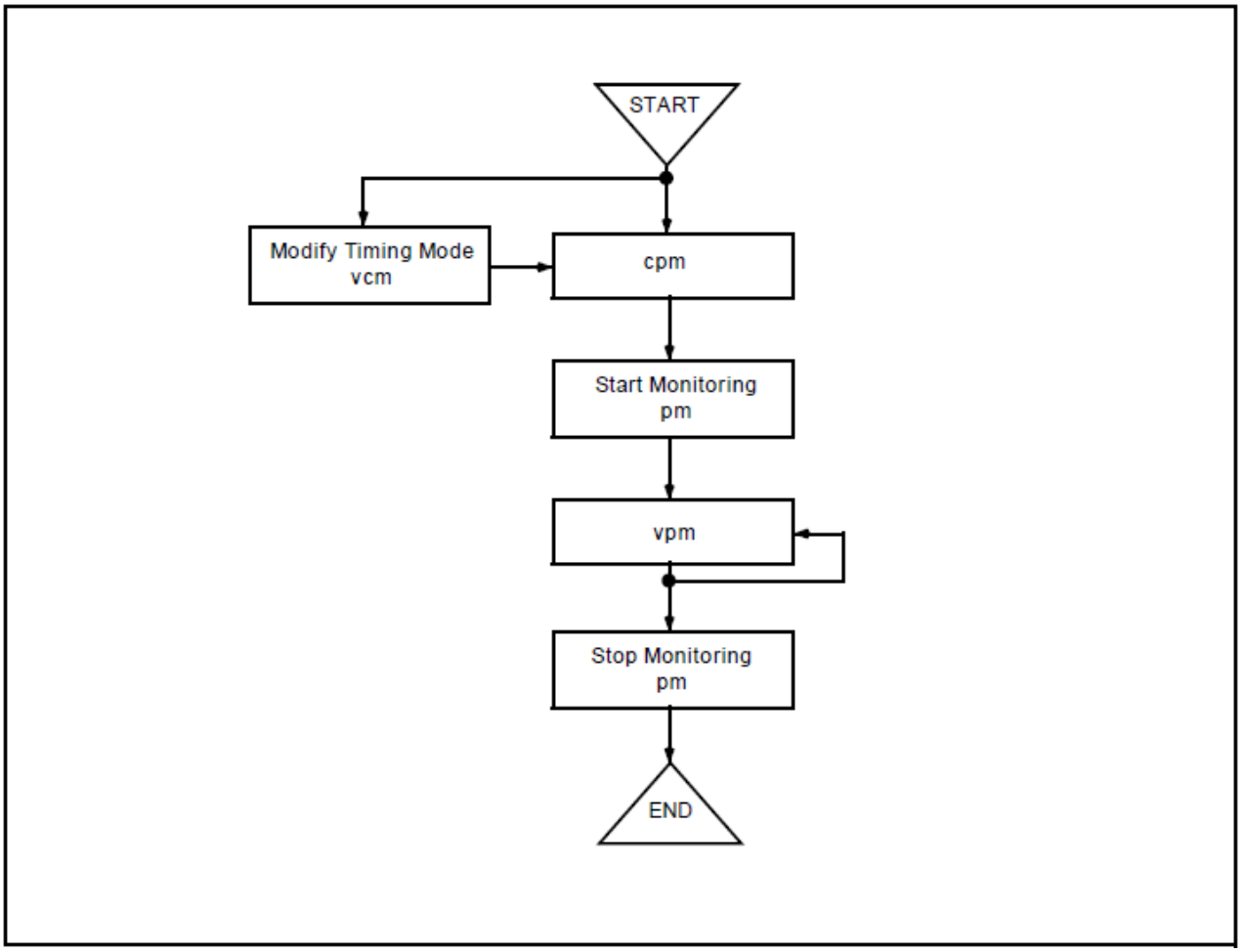
Frequency-Based Scheduler に関連する **`rtcp`** コマンドを実行する順序を図 5-1 に示します。

図5-1 rtcpコマンドの順序： Frequency-Based Scheduler



Performance Monitor に関連するコマンドを実行する順序を図 5-2 に示します。

図5-2 rtcpコマンドの順序： Performance Monitor



rtcpコマンドの使用

本項はリアルタイム・コマンド・プロセッサーでサポートされる全てのコマンドに関する参考情報を提供します。コマンドはヘルプ機能で説明している順番で提示しています。各コマンドについて、以下の情報が提供されます：

- コマンドの説明
- コマンドを入力するための書式
- 各引数に関する詳細な説明
- コマンドからの出力例

ats - Frequency-Based Schedulerにタイミング・ソースを接続

ats コマンドはタイミング・ソースを Frequency-Based Scheduler に接続またはエンド・オブ・サイクル・スケジューリングを指定します。後者の場合、現在のマイナー・サイクル中にスケジュールされた最後のプロセスがその処理を終了した時にスケジューリングが起動されます。

概要

ats -s *scheduler* {**-d** *device* | **-e**}

引数

-s *scheduler*	タイミング・ソースを接続するFrequency-Based Scheduler、または指定したエンド・オブ・サイクル・スケジューリングに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。これは任意の正の整数にすることが可能です。
-d *device*	指定されたスケジューラーのタイミング・ソースとして使用するリアルタイム・クロックまたはエッジ・トリガー割込みのパス名。各デバイスの種類に関連するフォームの詳細な情報について3章を参照して下さい。
-e	エンド・オブ・サイクル・スケジューリングを指定。この場合、次のマイナー・サイクルでのプロセスの実行は現在のマイナー・サイクルで実行するようにスケジュールされた最後のプロセスがそのサイクルで処理が終了する時に発生します。

表示

指定したタイミング・ソースが正常にスケジューラに接続された場合、またはエンド・オブ・サイクル・スケジューリングが正常に有効になった場合は以下のメッセージが表示されます。

```
Scheduler attached
```

chs - Frequency-Based Schedulerのアクセス権を変更

chs コマンドは Frequency-Based Scheduler に割り当てられたアクセス権を変更します。これを行うには、Frequency-Based Scheduler の作成者/所有者と同等の実効ユーザーID を持っている必要があります。

概要

chs -s *scheduler* [**-I** *permissions*] [**-G** *gid*] [**-U** *uid*]

引数

-s *scheduler*	アクセス権を変更するFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。
-I *permissions*	指定したスケジューラーに関連する操作で必要となるアクセス権。 引数*permissions*は3桁の8進数字を指定します(最初の桁は所有者に付与、2番目はグループに付与、3番目は他のユーザーに付与されたアクセス権を示します)。 スケジューラーに関連するアクセス権の変更のための8進数は**chmod**コマンド(**chmod(1)** のmanページを参照)の*mode*に指定するために使用されるものと同じです。デフォルトの**600**は読み取りと変更(書き込み)のアクセス権を所有者にのみ許可しています。
-G *gid*	選択されたFrequency-Based Schedulerの実効グループID。デフォルトの実行グループは現在のユーザーです。
-U *uid*	選択されたFrequency-Based Schedulerの実効ユーザーID。デフォルトは現在のユーザーです。

表示

スケジューラーに割り当てられたアクセス権が正常に変更された場合は以下のメッセージが表示されます。

```
Scheduler permissions changed
```

cs - Frequency-Based Schedulerを構成

cs コマンドは Frequency-Based Scheduler を構成します。

スケジューラーを生成するには、キーを指定する必要があり、それはスケジューラーに関連付けられるユーザーが選択した数値の識別子です。次もまた定義する必要があります。

- スケジューラーのメジャー・フレームを構成するマイナー・サイクルの数
- マイナー・サイクル中にスケジュールすることが可能なタスクの最大数
- スケジューラーに同時にスケジュールするタスクの最大数

Frequency-Based Scheduler はスケジューラー操作を実行するためのユーザーの機能を制御する 2 種類のアクセス権(読み取りと変更(書き込み))を使用します。読み取りアクセス権は問合せ操作を実行するのに必要となります。変更アクセス権は以下を必要とします：

- プログラムのスケジュール、削除、再スケジュール
- スケジューラーとタイミング・ソースを接続および分離
- スケジューリングを開始、停止、再開

スケジューラーを生成する時にアクセス権が割り当てられます。これらはファイルに関連するアクセス権が割り当てられるのと同じ方法で指定されます。アクセス権の指定については **chmod(1)** の man ページを参照して下さい。

cs コマンドを実行する時、以下の*両方*の条件を満足する場合は一意的な正の Frequency-Based Scheduler 識別子と対応するデータ構造体が指定されたキー用に生成されます：

- キーはまだFrequency-Based Scheduler識別子に関連付けられていない
- 既に構成したFrequency-Based Schedulerの数が、システムで許容されているスケジューラーの最大数よりも少ない

新たに生成された Frequency-Based Scheduler 識別子はターミナルに上に表示されます。

既に Frequency-Based Scheduler に関連付けれたキーを指定する時、以下の*全*ての条件を満足する場合は対応する Frequency-Based Scheduler 識別子がターミナル画面上に表示されます：

- **-C** *cycles/frame*引数で指定されたマイナー・サイクルの数が、既存のスケジューラーに関連付けられたマイナー・サイクルの数と一致
- **-M** *progs/cycle*引数で指定された最大値が、既存のスケジューラーに関連付けられたマイナー・サイクルあたりの最大プロセス数以下
- **-N** *progs/fbs*引数で指定された最大値が、既存のスケジューラーで許容される最大プロセス数以下

これらの条件を満たさない場合、エラー・メッセージが表示されます。

概要

cs -s *scheduler* **-C** *cycles/frame* **-M** *progs/cycle* **-N** *progs/fbs* [**-I** *permissions*] [**-R** *reset*]

引数

-s *scheduler*	生成したいFrequency-Based Scheduler用のキー。キーはスケジューラーに関連付けられるユーザーが選択した数値の識別子です。*scheduler*の値は任意の正の整数値が可能です。
-C *cycles/frame*	指定されたFrequency-Based Schedulerのフレームを構成するマイナー・サイクルの数。
-M *progs/cycle*	1マイナー・サイクル中にスケジュールして実行することが可能なプログラムの最大数。
-N *progs/fbs*	指定されたスケジューラーで同時にスケジュールすることが可能なプログラムの最大数。この値は、引数*cycles/frame*および*progs/cycle*で指定した値を掛けて得られる積以下である必要があります。
-I *permissions*	指定したスケジューラーに関連する操作で必要となるアクセス権。 引数*permissions*は3桁の8進数字を指定します(最初の桁は所有者に付与、2番目はグループに付与、3番目は他のユーザーに付与されたアクセス権を示します)。 スケジューラーに関連するアクセス権の変更のための8進数は**chmod**コマンド(**chmod(1)** のmanページを参照)の*mode*に指定するために使用されるものと同じです。デフォルトの**600**は読み取りと変更(書き込み)のアクセス権を所有者にのみ許可しています。
-R *reset*	指定されたスケジューラーに現在スケジュールされているプロセスを処理する方法。 *reset*引数の値は1、-1、0のいずれかが可能です。 0(デフォルト)を指定するとプロセスはスケジューラーに残します。1を指定するとスケジューラーからプロセスを削除しますが、実行を継続させます。-1を指定するとスケジューラーからプロセスを削除して終了します。

表示

スケジューラーが正常に構成された場合は、以下のメッセージが表示されます：

```
Scheduler 10 has FBS ID of 3
```

`Scheduler`

選択された Frequency-Based Scheduler のユーザー指定キーを示します。この値はリアルタイム・コマンド・プロセッサーの殆どで必要となることに注意することが重要です。

`FBS ID`

選択された Frequency-Based Scheduler の識別子を表す一意の正の整数値を示します。

dts - Frequency-Based Schedulerからタイミング・ソースを分離

dts コマンドは Frequency-Based Scheduler からタイミング・ソースを分離、またはエンド・オブ・サイクル・スケジューリングを無効にします。タイミング・ソースがリアルタイム・クロックである場合、**sc** (5-22 ページ)をコールして本ルーチンを起動する前にクロックを停止することを推奨します。

概要

dts -s *scheduler*

引数

-s *scheduler*	タイミング・ソースを分離するFrequency-Based Scheduler、または無効にするエンド・オブ・サイクル・スケジューリングに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

表示

タイミング・ソースがスケジューラーから正常に分離された、もしくはエンド・オブ・サイクル・スケジューリングが正常に無効になった場合は以下のメッセージが表示されます。

```
Scheduler detached
```

rms - Frequency-Based Schedulerを削除

rms コマンドは Frequency-Based Scheduler を削除します。本コマンドを実行する前に、スケジューラーのタイミング・ソースが分離されているまたはエンド・オブ・サイクル・スケジューリングが無効になっていることを確認する必要があります(**dts** コマンドの使用に関する情報については 5-15 ページの「dts – Frequency-Based Scheduler からタイミング・ソースを分離」を参照して下さい。)。

スケジューラーを削除するには、呼び出し元プロセスが Frequency-Based Scheduler の作成者/所有者と同等の実効ユーザーID を持っている必要があることに注意して下さい。

概要

rms -s *scheduler* [**-a**]

引数

-s *scheduler*	削除したいFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。
-a	指定されたスケジューラーに現在スケジュールされている全てのプロセスはスケジューラーから削除されて終了します。本オプションが指定されない場合、指定されたスケジューラ上に現在スケジュールされている全てのプロセスは削除されますが、実行は継続します。

表示

指定したスケジューラーが正常に削除された場合はコマンドは以下のメッセージを表示します。

```
Scheduler removed
```

svs - スケジューラーの構成を保存

svs コマンドは構成と選択した Frequency-Based Scheduler のスケジューリング・データをファイルに保存します。

本タスクを実行した場合、コマンド・ラインから入力された **rtcp** コマンドは指定された出力ファイルに保存されます。このファイルはその後入力ファイルとして利用することが可能です。**svs** で保存されたコマンドはスケジューラーの構成(**cs**)、プログラムのスケジュール(**sp**)、タイミング・ソースの接続(**ats**)、リアルタイム・クロックの設定(**stc**)に使用されます。

概要

svs -s *scheduler* **-d** *output_file_name*

引数

-s *scheduler*	構成とスケジューリング・データを保存したいFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。
-d *output_file_name*	構成データを保存したいファイルを特定する標準パス名称。これは最大1024文字のフル・パスまたは相対パスの名称にすることが可能です。

表示

スケジューラーの構成が正常にファイルに保存したかどうかのメッセージは表示しません。

vc - マイナー・サイクル/メジャー・フレームの総数を表示

vc コマンドは Frequency-Based Scheduler の現在のマイナー・サイクルとメジャー・フレームの総数値を表示します。これらの値はシミュレーションの進捗を判断するのに役立ちます。

概要

vc -s *scheduler*

引数

-s *scheduler*	現在のサイクルとフレームの総数を表示したいFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

表示

正常に実行された場合は、コマンドは以下の種類の情報を表示します：

```
Major frame = 65 Minor cycle = 25
```

`Major frame`

選択したスケジューラー上で実行中のシミュレーションに関する現在のメジャー・フレームの数を示します。

`Minor cycle`

選択したスケジューラー上で実行中のシミュレーションに関する現在のマイナー・サイクルの数を示します。

vs - スケジューラーの構成を表示

vs コマンドは Frequency-Based Scheduler に関連する情報を表示します。表示可能な情報は次を含みます：

- スケジューラーに関連付けられたキーおよび識別子
- メジャー・フレームあたりのマイナー・サイクルの数、マイナー・サイクルあたりのプログラムの最大数、スケジューラーあたりのプログラムの最大数
- スケジューラーの所有者と作成者のユーザーおよびグループID
- スケジューラーに割り当てられたアクセス権
- スケジューラ上の全プロセスに関するオーバーランの総数
- スケジューラーが実行または停止状態かどうかを表示
- システムでアクティブなCPUとPerformance Monitorが有効であるCPU
- スケジューラーに接続されているデバイスのパス名称

概要

vs -s *scheduler*

引数

-s *scheduler*	現在の情報を表示したいFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

表示

正常に実行された場合は、コマンドは構成およびステータス情報を表示します：

```
Scheduler 417 has FBS ID of 32768:            Cycles per frame = 101
Max programs per cycle = 10:         Max programs per FBS = 110
owner uid   = 9999:         owner gid   = 101
creator uid = 9999:         creator gid = 101:
total overruns = 0:         access mode = 600:          flags word = 1
active CPU mask = ' ----xxxx':      active PM CPU mask = ' ----x--x'
interrupt device name = /dev/rcim/rtc2
FBS is currently running
```

`Scheduler`

選択した Frequency-Based Scheduler のユーザー指定のキーを示します。

`FBS ID`

選択された Frequency-Based Scheduler の識別子を表す一意の正の整数値を示します。

`Cycles per frame`

選択したスケジューラー上のメジャー・フレームで構成するマイナー・サイクルの数を示します。

`Max programs per cycle`

選択したスケジューラー上のマイナー・サイクル内でスケジュールすることが可能なプログラムの最大数を示します。

`Max programs per FBS`

選択したスケジューラー上で同時にスケジュールすることが可能なプログラムの最大数を示します。

`owner uid`

スケジューラーの所有者のユーザーID を示します。

`owner gid`

スケジューラーの所有者のグループ ID を示します。

`creator uid`

スケジューラーの作成者のユーザーID を示します。

`creator gid`

スケジューラーの作成者のグループ ID を示します。

`total overruns`

選択したスケジューラー上の全プロセスに関するオーバーランの総数を示します。

`access mode`

選択したスケジューラーに割り当てられたアクセス権を表す 8 進数値を示します。

`flags word`

タイミング・ソースが選択したスケジューラーに接続されているかどうか、もしくはエンド・オブ・サイクル・スケジューリングが有効であるかどうかを示します。真の場合はこの領域に 1 を表示し、そうでなければ 0 を表示します。

`active CPU mask`

システムでアクティブな CPU のマスクを収容します。右端の位置は最初の論理 CPU に対応しています。文字 `x` は CPU がアクティブであることを表し、ダッシュ(-)はそうではないことを表します。

`active PM CPU mask`

Performance Monitor が有効である CPU のマスクを収容します。右端の位置は最初の論理 CPU に対応しています。文字 `x` は Performance Monitor が CPU で有効であることを表し、ダッシュ(-)はそうではないことを表します。

`interrupt device name`

タイミング・ソースが選択したスケジューラーに接続されている場合、この領域はデバイスのフル・パス名称を収容します。エンド・オブ・サイクル・スケジューリングが有効である場合、この領域は次を収容します： `EOC triggering`

`FBS is`

選択したスケジューラーの状態を示します。

ls - システム上の全スケジューラーを表示

ls コマンドはシステム上に構成された各 Frequency-Based Scheduler に関する情報を 1 行で表示します。

概要

ls

表示

本コマンドは構成された各スケジューラーに関する以下の情報を表示します：

```
ID    SCHEDULER   PERMISSIONS OWNER_UID   OWNER_GID   STATE      DEVICE
---   ----------  ----------- ---------   ----------  -------    ------------
0     37          --rw-rw-r-- fbsusr1     5309        running    /dev/rcim/rtc0
1     38          --rw-rw-r-- fbsusr2     5309        running    /dev/rcim/rtc1
2     39          --rw-rw-r-- fbsusr1     5309        stopped    /dev/rcim/rtc2
3     40          --rw-rw-r-- fbsusr3     5309        running    /dev/rcim/rtc3
```

rc - リアルタイム・クロックを開始

rc コマンドは選択した Frequency-Based Scheduler にタイミング・ソースとして指定したリアルタイム・クロック(**ats** コマンドの説明に関しては 5-11 ページの「ats – Frequency-Based Scheduler にタイミング・ソースを接続」を参照して下さい。)を開始します。最初にリアルタイム・クロックのカウントと分解能の値を **stc** コマンド(5-22 ページ)を実行して設定しておく必要があることに注意して下さい。

概要

rc -s *scheduler*

引数

-s *scheduler*	接続されたリアルタイム・クロックを開始したいFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

表示

リアルタイム・クロックが正常に開始された場合は以下のメッセージが表示されます：

`Clock started`

sc - リアルタイム・クロックを停止

sc コマンドは選択した Frequency-Based Scheduler にタイミング・ソースとして指定したリアルタイム・クロックを停止します。

概要

sc -s *scheduler*

引数

-s *scheduler*	接続されたリアルタイム・クロックを停止したいFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

表示

リアルタイム・クロックが正常に停止された場合は以下のメッセージが表示されます：

```
Clock stopped
```

stc - リアルタイム・クロックを設定

stc コマンドは選択した Frequency-Based Scheduler にタイミング・ソースとして指定したリアルタイム・クロックに関するカウントや持続時間の値を設定することでマイナー・サイクルの持続時間を規定します。

概要

sc -s *scheduler* [**-D** *clock_duraion*] **-O** *clock_count* [**-W** *tick_count*]

引数

-s *scheduler*	リアルタイム・クロックが接続されたFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。
-D *clock_duraion*	1クロック・カウントのマイクロ秒単位での持続期間。*clock_duraion*の値は次のいずれかにする必要があります：1, 10, 100, 1000, 10000。デフォルト値は10となります。
-O *clock_count*	マイナー・サイクルあたりのクロック・カウントの数。*clock_count*の値は2から65535の範囲が可能です。
-W *tick_count*	ウォッチドッグ・タイマーをサポートするデバイスに対して、*tick_count*はウォッチドッグ・タイマーがFBSの操縦を引き継ぐ前に除外することが可能な連続的な割り込みの総数としてドライバーに渡されます。本機能をサポートするドライバーにおいては、**ioctl**コマンドのWATCHDOGSET(_IOW('w',14, int))とWATCHDOGGET(_IOR('w',13, int))を受け入れる必要があります。

注意事項： *clock_count* に 1 を使用することは出来ませんが、*clock_duraion* の値と

等しいタイミング間隔は、次に低い *clock_duraion* の値と *clock_count* に 10 を指定することにより設定することが可能となります。例えば、**-D** 1000 **-O** 10 は 10,000 マイクロ秒のタイミング間隔をもたらします。

表示

リアルタイム・クロックが正常に設定された場合は以下のメッセージが表示されます：

```
Clock set
```

gtc - リアルタイム・クロックの設定を表示

gtc コマンドは選択した Frequency-Based Scheduler にタイミング・ソースとして指定したリアルタイム・クロックに関する現在のカウントと持続時間の値を表示します。クロックは事前に **stc** コマンド(5-22 ページ)を使って設定されている必要があります。

概要

gtc -s *scheduler*

引数

-s *scheduler* — リアルタイム・クロックが接続されたFrequency-Based Scheduler に関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

表示

リアルタイム・クロックが現在設定されている場合、本コマンドは以下の情報を表示します：

```
Clock count = 50: duration = 1000
```

start - Frequency-Based Schedulerのスケジューリングを開始

start コマンドは Frequency-Based Scheduler にスケジュールしたプロセスを開始します。本コマンドを実行した時、マイナー・サイクル、メジャー・フレーム、オーバーラン総数の値はゼロに設定されます。

本コマンドを実行する前に、スケジューラーにタイミング・ソースを接続するため、またはエンド・オブ・サイクル・スケジューリングを指定するために **ats** コマンドを実行しておく必要があります(**ats** コマンドの説明に関しては 5-11 ページの「ats – Frequency-Based Scheduler にタイミング・ソースを接続」を参照して下さい)。

スケジューラーへのタイミング・ソースとしてリアルタイム・クロックを指定した場合、**stc** コマンド(5-22 ページ)を使ってそのクロックを設定し、**rc** コマンド(5-21 ページ)を使ってそのクロックを開始するまではスケジューリングは開始されません。

タイミング・ソースとしてエッジ・トリガー割込みを指定した場合、正常にスケジューリングを開始するために割り込みは既に生成されている必要があります。

概要

start -s *scheduler*

引数

-s *scheduler*	スケジューリングを開始したいFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

表示

指定したスケジューラーのスケジューリングが正常に開始された場合、以下のメッセージが表示されます：

```
FBS started
```

resume - Frequency-Based Schedulerのスケジューリングを再開

resume コマンドはスケジューラーを停止した時と同じメジャー・フレーム、マイナー・サイクル、オーバーラン総数の値を使って選択した Frequency-Based Scheduler にスケジュールしたプロセスを再開します。

概要

resume -s *scheduler*

引数

-s *scheduler*	スケジューリングを再開したいFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

表示

指定したスケジューラーのスケジューリングが正常に再開された場合、以下のメッセージが表示されます：

```
FBS resumed
```

stop - Frequency-Based Schedulerのスケジューリングを停止

stop コマンドは選択した Frequency-Based Scheduler にスケジュールしたプロセスを停止します。

概要

stop -s *scheduler*

引数

-s *scheduler*	スケジューリングを停止したいFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

表示

指定したスケジューラーのスケジューリングが正常に停止された場合、以下のメッセージが表示されます：

```
FBS stopped
```

rmp - Frequency-Based Schedulerからプロセスを削除

rmp コマンドは Frequency-Based Scheduler からプロセスを削除します。以下のいずれかを指定して削除したいプロセスを特定して下さい：

- プロセスの名称とそれがスケジュールされているCPU
- プロセスのFrequency-Based SchedulerプロセスID
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based SchedulerプロセスID

NOTE

同じ CPU 上に複数回スケジュールされたプロセスを特定することが可能な唯一の方法は、その Frequency-Based Scheduler 識別子を指定することです。

概要

rmp -s *scheduler* {**-i** *fpid* | **-n** *proc_name* [**-c** *cpu_bias*]} [**-a**]

引数

-s *scheduler*
プロセスがスケジュールされているFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

-i *fpid*
削除するプロセスのFrequency-Based SchedulerプロセスID。この値は**sp**コマンド(5-32ページ)の実行により正常にプログラムがスケジュールされた時に表示されます。名称でプロセスを特定できない場合、この引数を指定する必要があります。

*fpid*のデフォルト値は-1です。デフォルト値を受け入れる場合、名称とCPUでプロセスを特定する必要があります。

-n *proc_name*
指定したスケジューラーから削除するプロセスを特定する標準パス名称。これは最大1024文字のフル・パスまたは相対パスの名称にすることが可能です。

この引数を指定しない場合、**-i** *fpid*引数を指定してFrequency-Based SchedulerプロセスIDを提供する必要があります。

-c *cpu_bias*
指定したスケジューラーから削除するプロセスを特定する**-n** *proc_name*引数の値と一緒に使用するプロセッサー。

*cpu_bias*の値は単一のCPU IDまたはCPU IDのリストにすることが可能です。CPU IDはゼロからCPUの数より1つ小さな値の範囲で、番号0は最初の論理CPUを表し、1は2番目を表します。

CPU IDのリストは並びもしくは数字の範囲を指定することが可能です(例えば、**`-c 1,3-5,7`**)。リスト内で項目を区切るのにコンマを使用する必要があることに注意して下さい。アスタリスク(*)を入力してCPU IDの全範囲を指定して下さい。

`-c` *cpu_bias*引数を指定しない場合、デフォルトのプロセッサーは**`rtcp`**が現在実行しているCPUになります。

`-a`

プロセスはスケジューラーから削除され終了します。本オプションが指定されない場合、プロセスはスケジューラーから削除されますが実行は継続されます。

表示

指定したプロセスが正常にスケジューラーから削除された場合、以下のメッセージが表示されます：

```
Process removed
```

rsp - プロセスを再スケジュール

rsp コマンドは Frequency-Based Scheduler にスケジュールされているプロセスのスケジューリング・パラメータを変更します。例えば、プロセスのスケジューリング・ポリシーや優先度の変更を望む可能性があります。同様にプロセスを実行するためにスケジュールされた頻度の変更を望む可能性もあります。しかしながら、スケジュールされた CPU を変更することは出来ません。

次の条件においては、呼び出し元プロセスの実効ユーザーID はターゲット・プロセス(スケジューリング・ポリシーと優先度が設定されているプロセス)の実効ユーザーID と一致している必要があります：

- 本コマンドを実行して、プロセスのスケジューリング・ポリシーを先入れ先出し(FIFO)またはラウンドロビン・ポリシーへ変更したい場合
- FIFOまたはラウンドロビン・ポリシーでスケジュールされたプロセスの優先度を変更したい場合
- プロセス単位の上限を超えてタイムシェアリング・ポリシーでスケジュールされたプロセスの優先度を上げたい場合

スケジューラーから削除するための **rmp** コマンドやスケジューリングを停止するための **stop** コマンドを最初に実行せずにプロセスを再スケジュールすることが可能です。

再スケジュールしたいプロセスを特定するには以下の方法のいずれかを使用して下さい：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based SchedulerプロセスIDを指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based SchedulerプロセスIDを指定

NOTE

同じ CPU 上に複数回スケジュールされたプロセスを特定することが可能な唯一の方法は、その Frequency-Based Scheduler 識別子を指定することです。

概要

rsp -s *scheduler* {**-i** *fpid* | **-n** *proc_name* [**-c** *cpu_bias*]} [**-f** *frequency*] \
[**-m** *start_cycle*] [**-b** *policy*] [**-p** *priority*] [**-o** *halt_flag*] [**-L** *soft_limit*] [**-T** *deadline*] \
[**-h** *halt_flag*] [**-r** *origin*]

引数

-s *scheduler*
: プロセスがスケジュールされているFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

-i *fpid*
: 再スケジュールするプロセスのFrequency-Based SchedulerプロセスID。この値は**sp**コマンド(5-32ページ)の実行した時に表示されます。名称でプロセスを特定できない場合、この引数を指定する必要があります。

*fpid*のデフォルト値は-1です。デフォルト値を使用する場合、名称とCPUでプロセスを特定する必要があります。

-n *proc_name*

再スケジュールするプロセスを特定する標準パス名称。これは最大1024文字のフル・パスまたは相対パスの名称にすることが可能です。この引数を指定しない場合、**-i** *fpid*引数を指定してFrequency-Based SchedulerプロセスIDを提供する必要があります。

-c *cpu_bias*

再スケジュールするプロセスを特定する**-n** *proc_name*引数の値と一緒に使用するプロセッサー。

*cpu_bias*の値は単一のCPU IDまたはCPU IDのリストにすることが可能です。CPU IDはゼロからCPUの数より1つ小さな値の範囲で、番号0は最初の論理CPUを表し、1は2番目を表します。

CPU IDのリストは並びもしくは数字の範囲を指定することが可能です(例えば、**-c 1,3-5,7**)。リスト内で項目を区切るのにコンマを使用する必要があることに注意して下さい。アスタリスク(*)を入力してCPU IDの全範囲を指定して下さい。

-c *cpu_bias*引数を指定しない場合、デフォルトのプロセッサーはリアルタイム・コマンド・プロセッサーが現在実行しているCPUになります。

-f *frequency*

指定したプロセスが各メジャー・フレームで起動される頻度。

頻度1は指定したプロセスが各マイナー・サイクルで起動されることを示し、頻度2は2つのマイナー・サイクルごとに1回起動されることを示します。

プロセスを起動させたい頻度を表すマイナー・サイクルの数を指定して下さい。*frequency*の値は1からスケジューラーのフレームを構成するマイナー・サイクルの数の範囲とすることが可能です。デフォルトは1です。

フレームあたりのマイナー・サイクルの総数は**cs**コマンド(5-13ページ)を実行することで定義されます。

-m *start_cycle*

指定したプロセスが各フレームで起動される最初のマイナー・サイクル。*start_cycle*の値は0からフレームあたりのマイナー・サイクルの総数-1の範囲とすることが可能です。デフォルトは0です。フレームあたりのマイナー・サイクルの総数は**cs**コマンド(5-13ページ)を実行することで定義されます。

-b *policy*

指定したプログラムに対するPOSIXスケジューリング・ポリシー：先入れ先出し(FIFO)スケジューリング・ポリシーを選ぶには**F**、ラウンドロビン・スケジューリング・ポリシーを選ぶには**R**、タイムシェアリング・スケジューリング・ポリシーを選ぶには**O**。

-b *policy*引数を指定しない場合、デフォルトのポリシーはタイムシェアリング・スケジューリング・ポリシーです。

注意事項： 本引数を指定することを推奨します。

-p *priority*
: 指定したプロセスのスケジューリング優先度。デフォルト値は0です。

  入力可能な優先度の値の範囲は*policy*引数の値に影響します。シェルから**run(1)**コマンドをオプションや引数を指定せずに起動して各ポリシー(**F**, **R**, **O**)に関連付けられた優先度の正当な範囲を確認して下さい。より高い数値はさらに有利なスケジューリング優先度に対応します。

-o *halt_flag*
: 指定したプロセスでフレーム・オーバーランが発生した場合にスケジューラーを停止させるべきかどうかを指示します。*halt_flag*の値は**halt**または**nohalt**のどちらかである必要があります。デフォルトは**nohalt**です。

-L *soft_limit*
: プロセスのソフト・オーバーラン上限。デフォルトは0です。ソフト・オーバーラン上限を非ゼロで設定済みのプロセスをソフト・オーバーラン上限を指定せずに再スケジュールした場合、プロセスのソフト・オーバーラン上限は0に設定されます。

-T *deadline*
: **fbswait(3)**に戻る前にタスクが実行するであろう最大時間を指定します。デッドライン制限を削除する**Clear**、もしくはマイクロ秒単位のデッドライン時間を示す正数にすることが可能です。デフォルト値は**Clear**です。

-h *halt_flag*
: 指定したプロセスでデッドライン違反が検出された場合にスケジューラーを停止すべきかどうかを指摘できるようにします。**halt**または**nohalt**を設定することが可能です。デフォルトは**halt**です。

-r *origin*
: デッドライン違反をテストする時に時間を測定するための開始ポイントを指示します。タスクがスケジューラされているサイクルの先頭から時間を測定するには**cycle**、タスクが**fbswait(3)**から抜け出して実行を開始した時に始まる時間を測定するには**task**を設定することが可能です。デフォルトは**cycle**です。

表示

指定したプロセスが正常に再スケジュールされた場合、以下が表示されます：

```
CPU  FPID  Prio.  Freq.  Start  Halt  SoftLimit  Ddln(us)/Origin  HaltOnDL  Process Name
 0     1  20/F     2        0   T        2           250/C          T      /home/jojo/wc
 0     0  53/F     2        0   T        0            None          -      /home/jojo/wc
```

CPU

指定したプロセスがスケジュールされている CPU の ID。

FPID

指定したプロセスの一意の Frequency-Based Scheduler プロセス ID。

この ID は **sp** コマンド(5-32 ページ)を使ってプログラムをスケジューラーにスケジュールした時にリアルタイム・コマンド・プロセッサーによって表示されます。

`Prio.`

指定したプロセスのスケジューリング優先度。

`Freq.`

指定したプロセスが各メジャー・フレームで起動するようにスケジュールされている頻度。

`Start`

指定したプロセスが各メジャー・フレームで起動するようにスケジュールされている最初のマイナー・サイクル。

`Halt`

「halt on overrun」フラグが指定したプロセスに設定されているかどうかを示します：Y (yes)または N (no)

`SoftLimit`

プロセスのソフト・オーバーラン上限。

`Ddln(us)/Origin`

スラッシュと起点コードで表示されるマイクロ秒単位のデッドライン時間。起点は、DL_CYCLE_RELATIVE が **C**、DL_TASK_RELATIVE が **T** です。

`HaltOnDl`

「Halt on deadline violation」フラグの値：**T**、**F** またはデッドラインが割り当てられていない場合は「-」。

`Process Name`

選択した Frequency-Based Scheduler にスケジュールされたフル・パスのプロセス名称。

sp - Frequency-Based Schedulerのプロセスをスケジュール

sp コマンドはプロセスを生成し、Frequency-Based Scheduler にそれをスケジュールします。

次の条件においては、呼び出し元プロセスの実効ユーザーID はターゲット・プロセス(スケジューリング・ポリシーと優先度が設定されているプロセス)の実効ユーザーID と一致している必要があります：

- 本コマンドを実行して、プロセスのスケジューリング・ポリシーを先入れ先出し(FIFO)またはラウンドロビン・ポリシーへ変更したい場合
- FIFOまたはラウンドロビン・ポリシーでスケジュールされたプロセスの優先度を変更したい場合
- プロセス単位の上限を超えてタイムシェアリング・ポリシーでスケジュールされたプロセスの優先度を上げたい場合

本コマンドを実行する時にプロセスの CPU バイアスを変更したい場合、呼び出し元プロセスの実ユーザー/実効ユーザーID が、CPU 割り当てを変更しようとしているプロセスの実ユーザー/保存済みユーザーID と一致している必要があります。

本コマンドを実行した時、リアルタイム・コマンド・プロセッサーは一意の Frequency-Based Scheduler プロセス ID を返します。その後、他のコマンドを実行する時にプロセスを指定するのに本 ID を使用することが可能です。

概要

sp -s *scheduler* **-n** *proc_name* [**-c** *cpu_bias*] [**-f** *frequency*] [**-m** *start_cycle*] [**-b** *policy*] \ [**-p** *priority*] [**-v** *parameter*] [**-o** *halt_flag*] [**-L** *soft_limit*] [**-T** *deadline*] [**-h** *halt_flag*] \ [**-r** *origin*] [**--** *arg1* [*arg2…*]]

引数

-s *scheduler*
: プロセスがスケジュールされることになるFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

-n *proc_name*
: スケジュールしたいプロセスを特定する標準パス名称。これは最大1024文字のフル・パスまたは相対パスの名称にすることが可能です。

-c *cpu_bias*
: 指定したプログラムのCPUバイアス。CPUバイアスはプログラムをスケジュールすることが可能なプロセッサーを決定します。

  *cpu_bias*の値は単一のCPU IDまたはCPU IDのリストにすることが可能です。CPU IDはゼロからCPUの数より1つ小さな値の範囲で、番号0は最初の論理CPUを表し、1は2番目を表します。

  CPU IDのリストは並びもしくは数字の範囲を指定することが可能です(例えば、**-c 1,3-5,7**)。リスト内で項目を区切るのにコンマを使用する必要があることに注意して下さい。アスタリスク(*)を入力してCPU IDの全範囲を指定して下さい。

-c *cpu_bias*引数を指定しない場合、デフォルトのプロセッサーはリアルタイム・コマンド・プロセッサーが現在実行しているCPUになります。

-f *frequency*

指定したプロセスが各メジャー・フレームで起動される頻度。

頻度1は指定したプロセスが各マイナー・サイクル毎で起動されることを示し、頻度2はマイナー・サイクル2回毎に1回起動されることを示します。

プロセスを起動させたい頻度を表すマイナー・サイクルの数を指定して下さい。*frequency*の値は1からスケジューラーのフレームを構成するマイナー・サイクルの数の範囲とすることが可能です。デフォルトは1です。フレームあたりのマイナー・サイクルの総数は**cs**コマンド(5-13ページ)を実行することで定義されます。

-m *start_cycle*

指定したプロセスが各フレームで起動するためにスケジュールされる最初のマイナー・サイクル。*start_cycle*の値は0からフレームあたりのマイナー・サイクルの総数-1の範囲とすることが可能です。デフォルトの値は0です。フレームあたりのマイナー・サイクルの総数は**cs**コマンド(5-13ページ)を実行することで定義されます。

-b *policy*

指定したプログラムに対するPOSIXスケジューリング・ポリシー：先入れ先出し(FIFO)スケジューリング・ポリシーを選ぶには**F**、ラウンドロビン・スケジューリング・ポリシーを選ぶには**R**、タイムシェアリング・スケジューリング・ポリシーを選ぶには**O**。

-b *policy*引数を指定しない場合、デフォルトのポリシーはタイムシェアリング・スケジューリング・ポリシーです。

注意事項： 本引数を指定することを推奨します。

-p *priority*

指定したプロセスのスケジューリング優先度。デフォルト値は0です。

優先度の値の範囲は*policy*引数の値に影響します。シェルから**run(1)**コマンドをオプションや引数を指定せずに起動して各ポリシー(**F, R, O**)に関連付けられた優先度の正当な範囲を確認して下さい。

より高い数値はさらに有利なスケジューリング優先度に対応します。

-v *parameter*

Frequency-Based Schedulerにスケジュールされているプロセスに渡される整数値。

*parameter*の値は32bitの10進数である必要があります。

-o *halt_flag*

指定したプログラムでフレーム・オーバーランが発生した場合にスケジューラーを停止させるべきかどうかを指示して下さい。

*halt_flag*の値は**halt**または**nohalt**のどちらかである必要があります。デフォルトは**nohalt**です。

-L *soft_limit*

プロセスのソフト・オーバーラン上限。デフォルトは0です。

-T *deadline*	**fbswait(3)**に戻る前にタスクが実行するであろう最大時間を指定します。デッドライン制限を削除する**Clear**、もしくはマイクロ秒単位のデッドライン時間を示す正数にすることが可能です。デフォルト値は**Clear**です。
-h *halt_flag*	指定したプロセスでデッドライン違反が検出された場合にスケジューラーを停止すべきかどうかを指摘できるようにします。**halt**または**nohalt**を設定することが可能です。デフォルトは**halt**です。
-r *origin*	デッドライン違反をテストする時に時間を測定するための開始ポイントを指示します。タスクがスケジューラされているサイクルの先頭から時間を測定するには**cycle**、タスクが**fbswait(3)**から抜け出して実行を開始した時に始まる時間を測定するには**task**を設定することが可能です。デフォルトは**cycle**です。
-- *arg1* [*arg2…*]	*argv[]*でスケジュールされたプロセスのコマンド・ラインに渡される引数。

表示

指定したプロセスが正常に Frequency-Based Scheduler にスケジュールされた場合は、以下の情報が表示されます：

```
fpid 199 assigned to process task02
```

`fpid`

指定したプロセスに割り当てられた一意の Frequency-Based Scheduler プロセス ID。

`process`

選択した Frequency-Based Scheduler にスケジュールされたプロセス名称のフル・パス。

vp - Frequency-Based Schedulerのプロセスを表示

vp コマンドは選択した Frequency-Based Scheduler において 1 つ以上のプロセッサー上にある FBS でスケジュールされた特定のプロセスまたは FBS でスケジュールされた全てのプロセスに関する情報を表示します。

スケジューラーおよびプロセスがスケジュールされている CPU を指定することで、指定したプロセッサーまたは全プロセッサー上の FBS でスケジュールされた全てのプロセスに関する情報を表示することが可能です。

FBS でスケジュールされた特定のプロセスに関する情報を表示したい場合、以下を指定することでプロセスを特定することが可能です：

- プロセスの名称とそれがスケジュールされているCPU
- プロセスのFrequency-Based SchedulerプロセスID
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based SchedulerプロセスID

NOTE

同じ CPU 上に複数回スケジュールされたプロセスを特定することが可能な唯一の方法は、その Frequency-Based Scheduler 識別子を指定することです。

各プロセスに対して表示される情報は次を含みます：

- プロセスが実行しているCPU
- Frequency-Based SchedulerプロセスID
- プロセスのスケジューリング・ポリシー
- 頻度(プロセスが各メジャー・フレームで起動する頻度を示すマイナー・サイクルの数)
- 開始ベース・サイクル(プロセスが各メジャー・フレームで起動するためにスケジュールされる最初のマイナー・サイクル)
- 「halt on overrun」フラグの値
- プロセスのソフト・オーバーラン上限
- 設定されているデッドライン時間
- デッドライン起点インジケーター
- 「halt on deadline violation」フラグの値
- プロセス名称のパス

概要

vp -s *scheduler* [**-n** *proc_name*] [**-i** *fpid*] [**-c** *cpu_bias*]

引数

-s *scheduler*	スケジューリング情報が表示されるFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。

Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

-n *proc_name*　スケジューリング情報を表示させる個別のプロセスを特定する標準パス名称。これは最大1024文字のフル・パスまたは相対パスの名称にすることが可能です。

-i *fpid*　スケジューリング情報を表示させる個別のプロセスのFrequency-Based SchedulerプロセスID。この値は**sp**コマンド(5-32ページ)を実行して正常にプログラムをスケジュールした時に表示されます。

*fpid*のデフォルト値は-1です。デフォルト値を受け入れる場合、名称とCPUでプロセスを特定する必要があります。

-c *cpu_bias*　スケジューリング情報を表示するプロセッサー。

*cpu_bias*の値は単一のCPU IDまたはCPU IDのリストにすることが可能です。CPU IDはゼロからCPUの数より1つ小さな値の範囲で、番号0は最初の論理CPUを表し、1は2番目を表します。

CPU IDのリストは並びもしくは数字の範囲を指定することが可能です(例えば、**-c 1,3-5,7**)。リスト内で項目を区切るのにコンマを使用する必要があることに注意して下さい。アスタリスク(*)を入力してCPU IDの全範囲を指定して下さい。

デフォルトのプロセッサーはリアルタイム・コマンド・プロセッサーが現在実行しているCPUになります。

表示

コマンドが正常に実行された場合、以下の情報が表示されます：

```
CPU  FPID  Prio.  Freq.  Start  Halt  SoftLimit  Ddln(us)/Origin  HaltOnDL  Process Name
 0      1  20/F      2      0    T         2          250/C          T      /home/jojo/wc
 0      0  53/F      2      0    T         0           None          -      /home/jojo/wc
```

CPU

それぞれのプロセスがスケジュールされている CPU の ID。

FPID

各プロセスの Frequency-Based Scheduler プロセス ID。ID は **sp** コマンド(5-32 ページ)を使ってスケジューラーにプログラムをスケジュールした時にリアルタイム・コマンド・プロセッサーにより表示されます。

`Prio.`

各プロセスのスケジューリング優先度。

`Freq.`

各プロセスが各メジャー・フレームで起動するようにスケジュールされている頻度。

`Start`

各プロセスが各メジャー・フレームで起動するようにスケジュールされている最初のマイナー・サイクル。

`Halt`

「halt on overrun」フラグが各プロセスに設定されているかどうかを示します：
Y (yes)または N (no)

`SoftLimit`

プロセスのソフト・オーバーラン上限。

`Ddln(us)/Origin`

スラッシュと起点コードで表示されるマイクロ秒単位のデッドライン時間。起点は、DL_CYCLE_RELATIVE が **`C`**、DL_TASK_RELATIVE が **`T`** です。

`HaltOnDl`

「Halt on deadline violation」フラグの値：**`T`**、**`F`** またはデッドラインが割り当てられていない場合は「-」。

`Process Name`

選択した Frequency-Based Scheduler にスケジュールされたフル・パスのプロセス名称。

cpm - Performance Monitorの値を消去

cpm コマンドは選択した Frequency-Based Scheduler の特定のプロセスまたは 1 つ以上のプロセッサー上の全てのプロセスに関する Performance Monitor の値を消去します。

スケジューラーおよびプロセスがスケジュールされている CPU を指定することで、指定したプロセッサーまたは全プロセッサー上の FBS でスケジュールされた全てのプロセスに関する値を消去することが可能です。

FBS でスケジュールされた特定のプロセスに関する Performance Monitor の値を消去したい場合、以下を指定することでプロセスを特定することが可能です：

- プロセスの名称とそれがスケジュールされているCPU
- プロセスのFrequency-Based Schedulerプロセス ID
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based Schedulerプロセス ID

概要

cpm -s *scheduler* [**-i** *fpid*] [**-n** *proc_name*] [**-c** *cpu_bias*]

引数

-s *scheduler*
プロセスがスケジュールされているFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。

Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

-i *fpid*
値を消去する個別のプロセスの一意のFrequency-Based Scheduler プロセスID。この値は**sp**コマンド(5-32ページ)を実行した時に表示されます。

*fpid*のデフォルト値は-1です。デフォルト値を受け入れる場合、名称とCPUでプロセスを特定する必要があります。

-n *proc_name*
値を消去する個別のプロセスを特定する標準パス名称。これは最大1024文字のフル・パスまたは相対パスの名称にすることが可能です。

-c *cpu_bias*
Performance Monitorの値を消去するプロセッサー。

*cpu_bias*の値は単一のCPU IDまたはCPU IDのリストにすることが可能です。CPU IDはゼロからCPUの数より1つ小さな値の範囲で、番号0は最初の論理CPUを表し、1は2番目を表します。

CPU IDのリストは並びもしくは数字の範囲を指定することが可能です(例えば、**-c 1,3-5,7**)。リスト内で項目を区切るのにコンマを使用する必要があることに注意して下さい。アスタリスク(*)を入力してCPU IDの全範囲を指定して下さい。

デフォルトのプロセッサーはリアルタイム・コマンド・プロセッサーが現在実行しているCPUになります。

表示

Performance Monitor の値が正常に消去された場合は、以下のメッセージが表示されます：

```
Performance monitor values cleared
```

NOTE

本コマンドはユーザーに指定された全てのプロセスのソフト・オーバーラン総数を消去します。

pm - Performance Monitorを開始/停止

pm コマンドは選択した Frequency-Based Scheduler の特定のプロセスまたは 1 つ以上のプロセッサー上の全てのプロセスに関する Performance Monitor を開始または停止します。

スケジューラーおよびプロセスがスケジュールされている CPU を指定することで、指定したプロセッサーまたは全プロセッサー上の FBS でスケジュールされた全てのプロセスに関する Performance Monitor を開始または停止することが可能です。

FBS でスケジュールされた特定のプロセスに関する Performance Monitor を開始または停止したい場合、以下を指定することでプロセスを特定することが可能です：

- プロセスの名称とそれがスケジュールされているCPU
- プロセスのFrequency-Based SchedulerプロセスID
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based SchedulerプロセスID

概要

pm -s *scheduler* [**-i** *fpid*] [**-n** *proc_name*] [**-c** *cpu_bias*] [**-P** *pm_flag*]

引数

-s *scheduler*
プロセスがスケジュールされているFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。

Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

-i *fpid*
Performance Monitorを開始または停止する個別のプロセスの一意のFrequency-Based SchedulerプロセスID。この値は**sp**コマンド(5-32ページ)を実行した時に表示されます。

*fpid*のデフォルト値は-1です。デフォルト値を受け入れる場合、名称とCPUでプロセスを特定する必要があります。

-n *proc_name*
Performance Monitorを開始または停止する個別のプロセスを特定する標準パス名称。これは最大1024文字のフル・パスまたは相対パスの名称にすることが可能です。

-c *cpu_bias*
Performance Monitorを開始または停止するプロセッサー。

*cpu_bias*の値は単一のCPU IDまたはCPU IDのリストにすることが可能です。CPU IDはゼロからCPUの数より1つ小さな値の範囲で、番号0は最初の論理CPUを表し、1は2番目を表します。

CPU IDのリストは並びもしくは数字の範囲を指定することが可能です(例えば、**-c 1,3-5,7**)。リスト内で項目を区切るのにコンマを使用する必要があることに注意して下さい。アスタリスク(*)を入力してCPU IDの全範囲を指定して下さい。

デフォルトのプロセッサーはリアルタイム・コマンド・プロセッサーが現在実行しているCPUになります。

-P *pm_flag* — Performance Monitorを開始または停止するかどうかを指示して下さい。*pm_flag*の値は**ON**または**OFF**はのどちらかである必要があります。デフォルトは**OFF**です。

表示

Performance Monitor が正常に開始された場合は、以下のメッセージが表示されます：

```
Performance monitoring enabled
```

Performance Monitor が正常に停止された場合は、以下のメッセージが表示されます：

```
Performance monitoring disabled
```

vcm - Performance Monitorのタイミング・モードを表示/修正

vcm コマンドは Performance Monitor のタイミング・モードを表示または修正します。タイミング・モードは Performance Monitor のタイミング値から割り込みを処理するのに費やされた時間を含めるまたは除外することを設定することが可能です。

割り込み時間が含まれている場合、プロセスのユーザーおよびシステム時間は、プロセスが現在処理を実行している時に割り込みを処理するのに費やした全ての時間を含めて生じた経過時間を合計します。割り込みを処理するのに費やした時間はプロセスのシステム時間に加算されます。

割り込み時間を除外している場合、プロセスのユーザーおよびシステム時間は、プロセスが現在処理を実行している時に割り込みを処理するのに費やした全ての時間を除外して生じた時間を合計します。

CAUTION

タイミング・モードは高分解能プロセス・タイミング機能がシステム全体で動作するモードです。これは全ての CPU 上で実行中の全てのプロセスに影響を及ぼします。

概要

vcm [**-t** *pm_mode*]

引数

-t *pm_mode*	割り込み時間をPerformance Monitorのタイミング値に含めるか、それとも除外するかを指定します。*pm_mode*の値は**in** (included)または**ex** (excluded)のどちらかである必要があります。

表示

Performance Monitor のタイミング・モードを表示する **vcm** コマンドが正常に実行された場合は、以下のメッセージのいずれかが表示されます：

```
PM timing mode includes interrupt times.
```

または

```
PM timing mode excludes interrupt times.
```

vpm - Performance Monitorの値を表示

vpm コマンドは選択した Frequency-Based Scheduler の 1 つのプロセスまたは 1 つ以上のプロセッサー上の全てのプロセスに関する Performance Monitor の値を表示します。

スケジューラーおよびプロセスがスケジュールされている CPU を指定することで、1 つのプロセッサーまたは全プロセッサー上の FBS でスケジュールされた全てのプロセスに関する値を表示します。

FBS でスケジュールされた特定のプロセスに関する Performance Monitor の値を表示したい場合、以下を指定することでプロセスを特定することが可能です：

- プロセスの名称とそれがスケジュールされているCPU
- プロセスのFrequency-Based SchedulerプロセスID
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based SchedulerプロセスID

概要

vpm -s *scheduler* [**-i** *fpid*] [**-n** *proc_name*] [**-c** *cpu_bias*] [**-x** *pm_output*]

引数

-s *scheduler*
プロセスがスケジュールされているFrequency-Based Schedulerに関連付けられた数字のキー。スケジューラーは事前に構成されている必要があります。

Frequency-Based Schedulerに関する情報については5-13ページの「cs – Frequency-Based Schedulerを構成」を参照して下さい。

-i *fpid*
Performance Monitorの値を表示する個別のプロセスの一意のFrequency-Based SchedulerプロセスID。この値は**sp**コマンド(5-32ページ)を実行した時に表示されます。

*fpid*のデフォルト値は-1です。デフォルト値を受け入れる場合、名称とCPUでプロセスを特定する必要があります。

-n *proc_name*
Performance Monitorの値を表示する個別のプロセスを特定する標準パス名称。これは最大1024文字のフル・パスまたは相対パスの名称にすることが可能です。

-c *cpu_bias*
Performance Monitorの値を表示するプロセッサー。

*cpu_bias*の値は単一のCPU IDまたはCPU IDのリストにすることが可能です。CPU IDはゼロからCPUの数より1つ小さな値の範囲で、番号0は最初の論理CPUを表し、1は2番目を表します。

CPU IDのリストは並びもしくは数字の範囲を指定することが可能です(例えば、**-c 1,3-5,7**)。リスト内で項目を区切るのにコンマを使用する必要があることに注意して下さい。アスタリスク(*)を入力してCPU IDの全範囲を指定して下さい。

デフォルトのプロセッサーはリアルタイム・コマンド・プロセッサーが現在実行しているCPUになります。

-x *pm_output*

表示する値の種類を指示します。4種類の値を指定することが可能です：average、minimum、maximum、all

反復またはサイクル数、最新時間、合計時間、平均時間、オーバーラン数を表示するには**av** (average)を指定。

反復回数、最小発生時の最小サイクル時間とマイナー・サイクルおよびメジャー・フレーム番号、最小発生時の最小フレーム時間とメジャー・フレーム番号を表示するには**mi** (minimum)を指定。

反復回数、最大発生時の最大サイクル時間とマイナー・サイクルおよびメジャー・フレーム番号、最大発生時の最大フレーム時間とメジャー・フレーム番号を表示するには**ma** (maximum)を指定。

average、minimum、maximumの値を表示するには**al** (all)を指定。

時間はミリ秒単位で報告されます。デフォルト値は**av** (average)です。

表示

average 値を選択してコマンドが正常に実行された場合、以下の情報が表示されます：

					OverRuns	
fpid	Iterations	TimeLast(us)	TotalTime(us)	Average(us)	Hard	Soft
199	480	1023	499200	1040	0	0
198	120	2013	240960	2008	1	2
197	240	1521	378960	1579	0	3

fpid

プロセスの一意の Frequency-Based Scheduler プロセス ID。本 ID は **sp** コマンド(5-32 ページ)を使ってスケジューラーにプログラムをスケジュールした時にリアルタイム・コマンド・プロセッサーで表示されます。

Iterations

最後に Performance Monitor の値が消去され Performance Monitor が有効になってから、プロセスが Frequency-Based Scheduler によって起動された回数。

`TimeLast(us)`

プロセスが最後に Frequency-Based Scheduler によって起動されてから **`fbswait`** を呼び出すまでに実行に費やしたマイクロ秒単位の時間。

`TotalTime(us)`

プロセスが全てのサイクルまたは反復で実行に費やしたマイクロ秒単位の累積時間。

`Average(us)`

個々のプロセスが全てのサイクルまたは反復で実行に費やしたマイクロ秒単位の平均時間。これらの値は「`TotalTime`」列で報告された値を「`Iterations`」列で報告された値で割って取得します。

`Overruns`

`Hard` 個々のプロセスが致命的なフレーム・オーバーランを引き起こした回数。

`Soft` 個々のプロセスが非致命的なフレーム・オーバーランを引き起こした回数。

`minimum` 値を選択してコマンドが正常に実行された場合、以下の情報が表示されます :

		Minimum Cycle		Minimum Frame	
fpid	Iterations	time(us)	Frame/Cycle	time(us)	Frame
199	30	1002	6/7	8013	17
198	30	1943	2/1	3995	22
197	30	1312	1/2	5314	11

`fpid`

個々のプロセスの一意の Frequency-Based Scheduler プロセス ID。本 ID は **`sp`** コマンド(5-32 ページ)を使ってスケジューラーにプログラムをスケジュールした時にリアルタイム・コマンド・プロセッサーで表示されます。

`Iterations`

最後に Performance Monitor の値が消去され Performance Monitor が有効になってから、個々のプロセスが Frequency-Based Scheduler によって起動された回数。

`Minimum Cycle time(us)`

プロセスがサイクルまたは反復で実行に費やしたマイクロ秒単位の最小時間。

`Minimum Cycle Frame/Cycle`

最小サイクル時間が発生したメジャー・フレームとマイナー・サイクルの番号。

`Minimum Frame time(us)`

プロセスがメジャー・フレーム中に実行に費やしたマイクロ秒単位の最小時間。

`Minimum Frame Frame`

最小フレーム時間が発生したメジャー・フレームの番号。

maximum 値を選択してコマンドが正常に実行された場合、以下の情報が表示されます：

		Maximum Cycle		Maximum Frame	
fpid	Iterations	time(us)	Frame/Cycle	time(us)	Frame
199	30	1303	2/4	9502	12
198	30	2201	7/5	4391	17
197	30	1917	9/6	7431	24

`fpid`

個々のプロセスの一意の Frequency-Based Scheduler プロセス ID。本 ID は **sp** コマンド(5-32 ページ)を使ってスケジューラーにプログラムをスケジュールした時にリアルタイム・コマンド・プロセッサーで表示されます。

`Iterations`

最後に Performance Monitor の値が消去され Performance Monitor が有効になってから、個々のプロセスが Frequency-Based Scheduler によって起動された回数。

`Maximum Cycle time(us)`

プロセスがサイクルまたは反復で実行に費やしたマイクロ秒単位の最大時間。

```
Maximum Cycle
Frame/Cycle
```

最大サイクル時間が発生したメジャー・フレームとマイナー・サイクルの番号。

```
Maximum Frame
time(us)
```

プロセスがメジャー・フレーム中に実行に費やしたマイクロ秒単位の最大時間。

```
Maximum Frame
Frame
```

最大フレーム時間が発生したメジャー・フレームの番号。

ex - リアルタイム・コマンド・プロセッサーを終了

ex コマンドはリアルタイム・コマンド・プロセッサーを対話モードで使用している場合のみ使用されます。これはコマンド・プロセッサーを終了してシェルに戻ります。

概要

ex

コマンドが正常に実行された場合はシステム・コマンド・プロンプトが表示されます。

he - ヘルプ情報を表示

he コマンドはリアルタイム・コマンド・プロセッサーに関する以下の種類のヘルプ情報を表示します：

- 全コマンドの一覧と簡単な説明
- 特定コマンドの説明と書式
- 全コマンド引数の一覧と説明

概要

he [*command*] [**option** | **op2**]

引数

command	情報を取得したいコマンド。
option	コマンド引数の初期画面が表示されます。
op2	コマンド引数の2番目の画面が表示されます。

表示

引数なしで **he** コマンドを指定した場合、簡単な説明と一緒に全コマンドの一覧が表示されます。5-5 ページの画面 5-1 を参照して下さい。

option 引数を付けて **he** コマンドを指定した場合、**rtcp** コマンド引数の初期画面が表示されます。5-6 ページの画面 5-2 を参照して下さい。

op2 引数を付けて **he** コマンドを指定した場合、**rtcp** コマンド引数の 2 番目の画面が表示されます。5-7 ページの画面 5-3 を参照して下さい。

command 引数を付けて **he** コマンドを指定した場合、指定したコマンドの説明とコマンドを入力するための書式が表示されます。例えば、**he chs** は以下を表示します :

```
Change FBS permissions

rtcp chs -s scheduler -I permissions [-G gid] [-U uid]
```

6章

Cライブラリ・インターフェース

概要 6-1
プログラムのコンパイルとリンク 6-1
Big-SMP FBSインターフェース 6-2
Frequency-Based Schedulerルーチン 6-4
ルーチンの要約 6-4
Cライブラリ・コールの順序 6-6
Frequency-Based Schedulerルーチンの使用 6-7
fbsaccess – Frequency-Based Schedulerのアクセス権を変更 6-7
fbsattach – Frequency-Based Schedulerにタイミング・ソースを接続 6-8
fbsavail – Frequency-Based Schedulerが構成されているかどうかを問合せ 6-9
fbsconfigure – Frequency-Based Schedulerを構成 6-10
fbscycle – マイナー・サイクル/メジャー・フレームの総数を返答 6-13
fbsdetach – Frequency-Based Schedulerからタイミング・ソースを分離 6-14
fbsdir – スケジューラー・キーのリストを返答 6-15
fbsgetpid – スケジュール済みプロセスのプロセスIDを返答 6-16
fbsgetrtc – リアルタイム・クロックの現在値を取得 6-17
fbsid – キーに対するFrequency-Based Scheduler識別子を返答 6-18
fbsinfo, fbsinfo_big – Frequency-Based Schedulerの情報を返答 6-19
fbsintrpt – スケジューリングを開始/停止/再開 6-22
fbsremove – Frequency-Based Schedulerを削除 6-23
fbsresume – Frequency-Based Schedulerのスケジューリングを再開 6-24
fbsrunrtc – リアルタイム・クロックを開始/停止 6-25
fbsschedself – Frequency-Based Schedulerに呼び出し元プロセスを追加 6-26
fbssetrtc – リアルタイム・クロックを設定 6-28
fbstrig – スリープ中のFrequency-Based Schedulerプロセスを実行可能に遷移 . . . 6-29
fbswait – Frequency-Based Schedulerを待機 6-30
nametopid, namepid, nametopid_big, namepid_big – 指定したプロセス名称のプロセスIDを返答 6-31
pgmremove, pgmremove_big – Frequency-Based Schedulerからプロセスを削除 . . . 6-33
pgmtrigger – Frequency-Based Schedulerのプロセスを起動 6-35
sched_fbsqry, sched_fbsqry_big – Frequency-Based Schedulerのプロセスを問合せ . 6-36
sched_pgm_deadline_query, sched_pgm_deadline_query_big – プロセスに割り当て済みデッドライン時間を問合せ 6-39
sched_pgm_deadline_test, sched_pgm_deadline_test_big – デッドライン違反の有無をテスト 6-42
sched_pgm_set_deadline, sched_pgm_set_deadline_big – プロセス・デッドライン時間を設定または消去 6-45
sched_pgm_set_soft_overrun_limit, sched_pgm_set_soft_overrun_limit_big – ソフト・オーバーラン上限を設定 6-48
sched_pgmadd, sched_pgmadd_big – Frequency-Based Schedulerにプロセスをスケジュール 6-51
sched_pgmadd_args, sched_pgmadd_args_big – Frequency-Based Schedulerに引数を付けてプロセスをスケジュール 6-54

sched_pgmadd_attr, sched_pgmadd_attr_big – Frequency-Based Schedulerに引数と属性を付けてプロセスをスケジュール 6-58
sched_pgmqry, sched_pgmqry_big – プロセスを問合せ 6-61
sched_pgmresched, sched_pgmresched_big – プロセスを再スケジュール 6-65
Performance Monitorルーチン 6-69
ルーチンの要約 6-69
Cライブラリ・コールの順序 6-69
Performance Monitorルーチンの使用 6-70
pmclrpgm, pmclrpgm_big, – プロセスの値を消去 6-71
pmclrtable, pmclrtable_big – プロセッサーの値を消去 6-73
pmmonitor, pmmonitor_big – プロセッサーのPerformance Monitorを開始/停止 . . . 6-75
pmprogram, pmprogram_big – プロセスのPerformance Monitorを開始/停止 6-77
pmqrycpu, pmqrycpu_big – 選択したプロセッサーの値を問合せ 6-79
pmqrylist – プロセスのリストの値を問合せ 6-83
pmqrypgm, pmqrypgm_big – 選択したプロセスの値を問合せ 6-85
pmqrytimer – Performance Monitorのモードを問合せ 6-88
pmselect – Performance Monitorのモードを選択 6-89

6章

Cライブラリ・インターフェース

本章では Frequency-Based Scheduler と Performance Monitor への C ライブラリ・インターフェースについて説明します。ライブラリ情報、スケジューラーおよび Performance Monitor の呼び出し順序、各々のライブラリ・ルーチンの詳細を提供します。

概要

Frequency-Based Scheduler と Performance Monitor に関連付けられた機能へのアクセスは C 言語で書かれたアプリケーションから呼び出すことが可能なルーチンのライブラリを通して提供されます。

各ルーチンについて以下の情報が本章内で提供されます：

- ルーチンの説明
- ルーチンの概要
- 各パラメータの詳細な説明
- 戻り値

FBS でスケジューリング中にルーチンを呼び出す順序の図を含む個々のルーチンに関する情報は 6-4 ページから始まります。Performance Monitor に関する同じ情報が 6-69 ページから提供されます。

Frequency-Based Scheduler と Performance Monitor への C ライブラリ・インターフェースの使用を解説するプログラム例が付録 C で提供されます。

プログラムのコンパイルとリンク

C プログラムを静的にリンクする場合、以下のライブラリが必要となります：

/usr/lib/libccur_fbsched.a または**/usr/lib64/libccur_fbsched.a**
/usr/lib/libccur_rt.a または**/usr/lib64/libccur_rt.a**

C プログラムを動的にリンクする場合、以下のライブラリが必要となります：

/usr/lib/libccur_fbsched.so または**/usr/lib64/libccur_fbsched.so**
/usr/lib/libccur_rt.so または**/usr/lib64/libccur_rt.so**

C プログラムをコンパイルしてリンクするには、コマンド・ライン命令は次のとおりです：

gcc [*options…*] *source_file*.**c -lccur_fbsched -lccur_rt**

コンパイルとリンクの手順に関する追加情報については、**gcc(1)** と **ld(1)** の man ページを参照して下さい。

Big-SMP FBSインターフェース

歴史的に FBS API は CPU マスクとして 32bit 整数を使用していました。これは FBS が Big-SMP プラットフォームでの使用を不可能にしています。(つまり、値が 32 以上になる複数の CPU ID を持つプラットフォームにおいて)

従って、新しい FBS API が設計されました。古い API から新しい API へアプリケーションの変更を支援するため、新しい FBS API は可能な限り古いものと一致させました。この新しい API は古いものを置換しないことに注意して下さい。プラットフォームの最大値が 32 未満である限りは、新旧の FBS API の両方が互換的に使用することが可能です。

新しい FBS API を生成するために使用された大まかな変換ルールは以下のようになります：

1. いたるところで CPU マスク(**`cpuset(3)`**で定義されているように CPU セットへのポインター)として整数が使用されていた古い API が代わりに使用されます。
2. 直接的または間接的に整数のビットマスクを使用する全ての古い FBS API 関数については、新しい Big-SMP バージョンの関数は古いものと同じ名称に「`_big`」を末尾に加えて生成しました。例えば、**`fbsinfo_big(3)`**は**`fbsinfo(3)`**の Big-SMP 発展型です。
3. 整数の CPU マスク領域を持つ全ての古い FBS API データ構造体については、新しい Big-SMP 変異体は古いものと同じ名称で「`_big`」がその名称のどこかに挿入されています。例えば、「`struct pgm2_ds`」の Big-SMP 変異体は「`struct pgm2_big_ds`」です。
4. 全ての新しい FBS API データ構造体については、各 `cpuset` 領域は古い整数の CPU マスク領域と同じ名称で、「`cpu`」の文字が「`cpuset`」または「`cpuset_`」に置き換えられます。例えば、「`struct fbsinfo_ds`」の「`pm_cpuactive`」領域が「`struct fbsinfo_big_ds`」の「`pm_cpuset_active`」領域になります。
5. 直接的または間接的に整数の CPU マスクを参照しない FBS API データ構造体もしくは関数に変更はありません。例えば、「`struct pgmqry_ds`」はその中に CPU マスクはないので、**`pmqrypgm(3)`**と Big-SMP 変異体 **`pmqrypgm_big(3)`**の両方で使用されます。

全体として、FBS API の新しい関数が 21 個と新しいデータ構造体が 9 個あります。

少しでも簡単にアプリケーションの変更を行うには、以下の `cpuset` 拡張が可能です：

1. `cpuset` が出力専用パラメータである FBS API 内のそれらの部分については、`cpuset` ポインターが `NULL` の場合は `cpuset` は返されません。これはアプリケーションが返された `cpuset` を使用していないことが分かっている状況(大抵の場合でその可能性があります)においては便利です。
2. `cpuset` が入力専用パラメータである FBS API 内のそれらの部分については、`cpuset` へのポインターは実際の `cpuset` へのポインターの代わりに以下のいずれかの特殊な値になる可能性があります。それらの特殊なポインターは以下となります：

   `FBS_CPUSET_ALLCPUS`　　フル・セット。

`FBS_CPUSET_CURCPU`	単一CPU ID(呼び出し元CPU)を持つcpuset。
`FBS_CPUSET_ONECPU(`*cpu*`)`	単一CPU ID(CPU ID *cpu*を指定)を持つcpuset。

3. `cpuset` ポインターが実際の `cpuset` を指していても期せずしてそれが空のセットである場合、その空のセットは `FBS_CPUSET_CURCPU` と同じように扱われます。これはCPU マスクに値 0(そうでなければ規則違反)が設定されている場合に古い Small-SMP インターフェースで発生することを反映します。
4. `cpuset` 引数が入出力パラメータである場合、そのポインターは上記の特殊な値のいずれかをやはり設定することが可能ですが、その場合は出力の `cpuset` は返されません。

Frequency-Based Schedulerルーチン

Frequency-Based Scheduler ルーチンはスケジューラーの主要な機能へのアクセスを提供します。それらは以下のような基本的な操作を実行することを可能にします：

- スケジューラーを構成
- スケジューラーにプログラムをスケジュール
- スケジューラーにタイミング・ソースを設定および接続
- スケジューラーのスケジュールを開始、停止、再開
- スケジュールされたプロセスに関する情報を取得
- スケジュールされたプロセスを再スケジュールおよび削除
- タイミング・ソースを分離
- スケジューラーの削除

ルーチンの要約

Frequency-Based Scheduler ルーチンが表 6-1 に要約されています。各ルーチンに関する完全な情報は「Frequency-Based Scheduler ルーチンの使用」項で提供されます。

表6-1 Frequency-Based SchedulerのCライブラリ・ルーチン

ルーチン	ページ	説明
fbsaccess	6-7	Frequency-Based Schedulerのアクセス権を変更
fbsattach	6-8	Frequency-Based Schedulerにタイミング・ソースを接続
fbsavail	6-9	FBSがシステムに構成されているかを問合せ
fbsconfigure	6-10	Frequency-Based Schedulerを構成
fbscycle	6-13	マイナー・サイクル/メジャー・フレームの総数を返答
fbsdetach	6-14	Frequency-Based Schedulerからタイミング・ソースを分離
fbsdir	6-15	スケジューラー・キーのリストを返答
fbsgetpid	6-16	スケジュール済みプロセスのプロセスIDを返答
fbsgetrtc	6-17	リアルタイム・クロックの値を取得
fbsid	6-18	キーに対するFrequency-Based Scheduler識別子を返答
fbsinfo fbsinfo_big	6-19	Frequency-Based Schedulerの情報を返答
fbsintrpt	6-22	スケジューリングを開始/停止/再開
fbsremove	6-23	Frequency-Based Schedulerを削除

(次ページに続く)

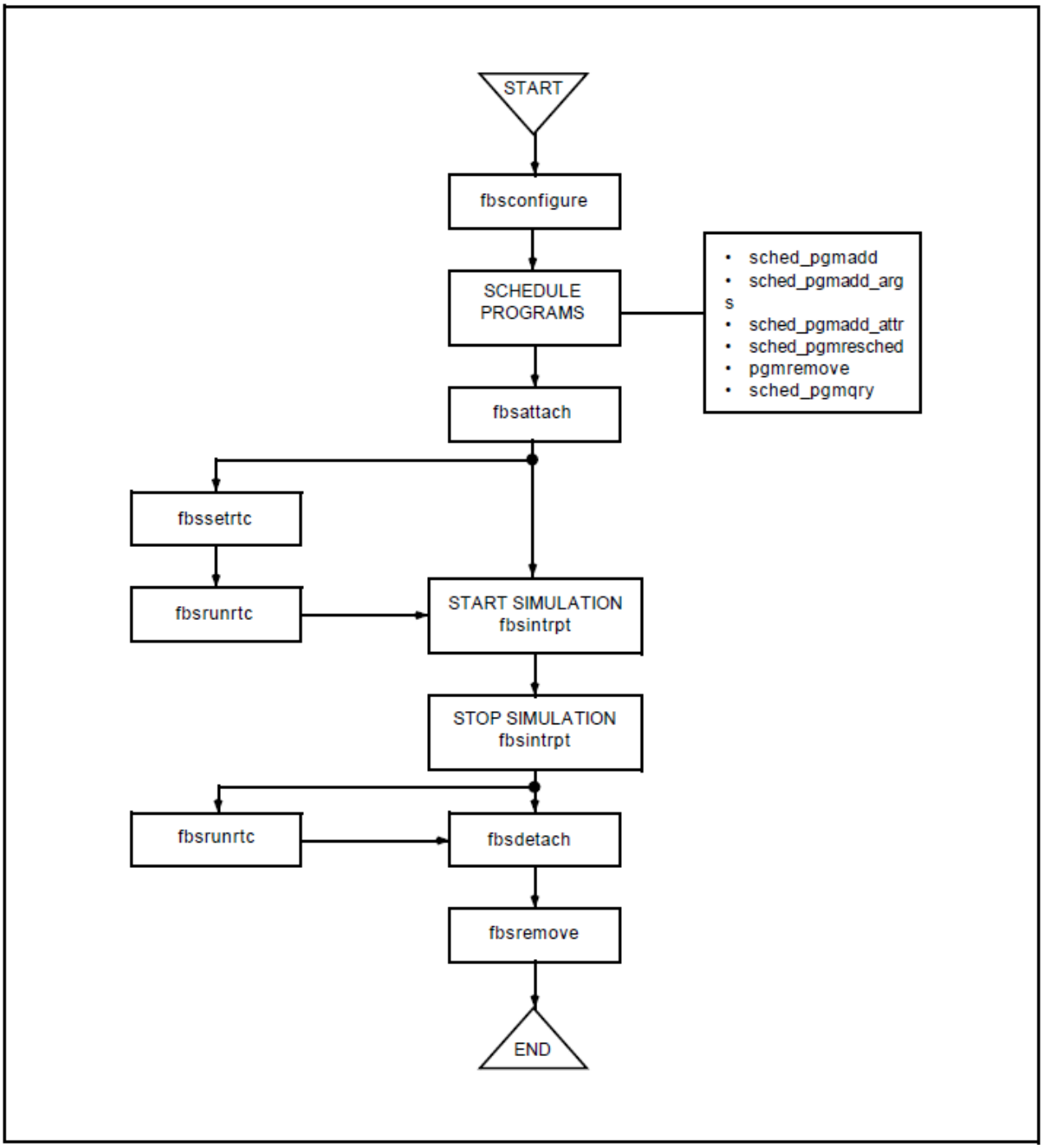
表6-1 Frequency-Based SchedulerのCライブラリ・ルーチン (続き)

ルーチン	ページ	説明
fbsresume	6-24	Frequency-Based Schedulerのスケジューリングを再開
fbsrunrtc	6-25	リアルタイム・クロックを開始/停止
fbsschedself	6-26	Frequency-Based Schedulerに呼び出し元プロセスを追加
fbssetrtc	6-28	リアルタイム・クロックを設定
fbstrig	6-29	スリープ中のFBSでスケジュールされたプロセスを実行可能に遷移
fbswait	6-30	Frequency-Based Schedulerを待機
namepid namepid_big	6-31	指定したプロセス名称のプロセスIDを返答
nametopid nametopid_big	6-31	指定したスケジューラーの指定したプロセス名称のプロセスIDを返答
pgmremove pgmremove_big	6-33	Frequency-Based Schedulerからプロセスを削除
pgmtrigger	6-35	Frequency-Based Schedulerのプロセスを起動
sched_fbsqry sched_fbsqry_big	6-36	Frequency-Based Schedulerのプロセスを問合せ
sched_pgm_deadline_query sched_pgm_deadline_query_big	6-39	割り当て済みデッドライン時間を問合せ
sched_pgm_deadline_test sched_pgm_deadline_test_big	6-42	デッドライン違反についてプロセスを確認
sched_pgm_set_deadline sched_pgm_set_deadline_big	6-45	プロセス・デッドライン時間を設定または消去
sched_pgm_set_soft_overrun_limit sched_pgm_set_soft_overrun_limit_big	6-48	ソフト・オーバーラン上限を設定
sched_pgmadd sched_pgmadd_big	6-51	Frequency-Based Schedulerにプロセスをスケジュール
sched_pgmadd_args sched_pgmadd_args_big	6-54	Frequency-Based Schedulerに引数を付けてプロセスをスケジュール
sched_pgmadd_attr sched_pgmadd_attr_big	6-58	Frequency-Based Schedulerに引数と属性を付けてプロセスをスケジュール
sched_pgmqry sched_pgmqry_big	6-61	プロセスを問合せ
sched_pgmresched sched_pgmresched_big	6-65	プロセスを再スケジュール

Cライブラリ・コールの順序

アプリケーション・プログラムからルーチンを呼び出す可能性のあるおおよその順番を図 6-1 に示します。

図6-1 Cライブラリ・コールの順序：Frequency-Based Scheduler



Frequency-Based Schedulerの使用

これ以降は、**libccur_fbsched** ライブラリ内に含まれる Frequency-Based Scheduler ルーチンをアルファベット順で示します。

fbsaccess - Frequency-Based Schedulerのアクセス権を変更

本ルーチンは選択した Frequency-Based Scheduler に割り当てられたアクセス権を変更します。アクセス権は Frequency-Based Scheduler の所有者/生成者と同じ実効ユーザーID を持つプロセスによってのみ変更可能なことに注意することが重要です。

概要

```
#include <fbsched.h>

int fbsaccess （fbs_id, uid, gid, permissions）
int fbs_id, uid, gid, permissions;
```

パラメータ

fbs_id — Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

uid — 指定したFrequency-Based Schedulerの実効ユーザーIDを表す整数値。

gid — 指定したFrequency-Based Schedulerの実効グループIDを表す整数値。

permissions — 指定したFrequency-Based Schedulerに関連付けられたアクセス権を設定するために使用されるビット・パターン。ビット・パターンと対応するアクセス権を表6-2に示します。

表6-2 Frequency-Based Schedulerのアクセス権

ビット・パターン	アクセス権
400	ユーザーによる読み取り
200	ユーザーによる変更
060	グループによる読み取り、変更
006	その他ユーザーによる読み取り、変更

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために errno が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbsaccess(3)** の man ページを参照して下さい。

fbsattach - Frequency-Based Schedulerにタイミング・ソースを接続

本ルーチンはタイミング・ソースを Frequency-Based Scheduler に接続、またはエンド・オブ・サイクル・スケジューリングを指定します。タイミング・ソースはリアルタイム・クロックまたはエッジ・トリガー割込みデバイスにすることが可能です。

概要

```
#include <fbsched.h>

int fbsattach (fbs_id, devname)
int fbs_id;
char *devname;
```

パラメータ

fbs_id

Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

devname

NULL文字列または指定したスケジューラーのタイミング・ソースとして使用するデバイスのパス名称。

*devname*がNULL文字列である場合、エンド・オブ・サイクル・スケジューリングが指定されます。つまり、現在のマイナー・サイクルで実行するようにスケジュールされた最後のプロセスがそのサイクルで実行を終了した時、次のマイナー・サイクルでのプロセスの実行が発生します。

*devname*がパス名称を含む場合、リアルタイム・クロックまたはエッジ・トリガー割込みを参照することが可能です。タイミング・ソースのデバイス・ファイルに関する情報については3章を参照して下さい。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbsattach(3)** の man ページを参照して下さい。

fbsavail - Frequency-Based Schedulerが構成されているかどうかを問合せ

本ルーチンは Frequency-Based Scheduler が構成されているかどうかを判定するために現在実行中のカーネルに問い合わせます。

概要

```
#include <fbsched.h>

int fbsavail ()
```

戻り値

戻り値 1 は FBS が構成されていることを示します。戻り値 0 は FBS が構成されていないことを示します。

fbsconfigure - Frequency-Based Schedulerを構成

本ルーチンは Frequency-Based Scheduler を構成、または既に構成されているスケジューラーの構成詳細を取得します。

以下のように `fbsconfig_ds` 構造体の *cycles* 領域に割り当てられた値を使って実行する機能を指定して下さい：

- *cycles*に指定した値が0ではない、かつ既存のFrequency-Based Schedulerの数がシステムに課せられた上限(デフォルト=10)未満である場合、Frequency-Based Schedulerが生成され識別子が*`fbs_id`*領域に返されます。

  `fbsconfig_ds`構造体の*reset*領域に割り当てられた値は、*cycles*が0ではなくスケジューラーが既に存在する場合に必要となる動作を更に限定するために使用することが可能です。

- *cycles*の値が0である場合、Frequency-Based Schedulerの現在の構成が`fbsconfig_ds`構造体に返されます。ユーザー指定の*key*領域はFrequency-Based Schedulerを特定するために使用することが可能です。

概要

```
#include <fbsched.h>

int fbsconfigure (fbs_buf)
struct fbsconfig_ds {
    int key;
    int cycles;
    int progs;
    int max;
    int reset;
    int configflg;
    int fbs_id;
} *fbs_buf;
```

パラメータ

Frequency-Based Scheduler を生成するには、説明に従い以下のパラメータを指定する必要があります。

fbs_buf — 構成したいFrequency-Based Schedulerの情報を含む`fbsconfig_ds`構造体。各コンポーネントに指定する情報の種類を以下に示します。

- *key* — 生成するFrequency-Based Schedulerを後で特定するために使用することが可能な整数値。ゼロは有効なキーの値ではありませんが、キーが使用されていないことを示すために割り当てることが可能です。ここで返される*fbs_id*の値は常にFrequency-Based Schedulerを参照するために使用することが可能であるため、これはオプションの識別子です。
- *cycles* — 指定したスケジューラーのフレームを構成するマイナー・サイクルの数を示す整数値。

progs 1マイナー・サイクル中にスケジュールして実行することが可能なプログラムの最大数を表す整数値。

max 指定したスケジューラーで同時にスケジュール可能なプログラムの最大数を表す整数値。この値は*cycles*および*progs*パラメータで指定した値を掛けて得られる積以下である必要があります。

reset 指定したスケジューラーに現在スケジュールされているプロセスをスケジューラーが再構成される前に終了するかどうかを表す整数値。許容値および対応する結果は次のとおりです：

<0 指定したスケジューラーに現在スケジュールされている全てのプロセスを終了して削除
0 指定したスケジューラーに現在スケジュールされている全てのプロセスを無視
>0 指定したスケジューラーに現在スケジュールされている全てのプロセスを削除

configflg 指定したスケジューラーに割り当てられるアクセス権を表す整数値。

fbs_id **fbsconfigure**によって返される一意の正の整数値で指定したFrequency-Based Schedulerの識別子を表します。この識別子はFrequency-Based SchedulerとPerformance Monitorの殆どのライブラリ・ルーチンで必要になることに注意することが重要です。

既存の Frequency-Based Scheduler に関する情報を取得するには、説明に従い以下のパラメータを指定する必要があります。

fbs_buf 既存のFrequency-Based Schedulerに関する情報が返される`fbsconfig_ds`構造体。各コンポーネントに指定または返される情報の種類を以下に示します。

key スケジューラーが生成された時に指定したユーザー指定の値。この値は構成情報が返されるFrequency-Based Schedulerを特定するために使用することが可能です。

cycles 指定したスケジューラーの現在の構成情報が返されることを表す整数値は0。指定したスケジューラーのフレームを構成するマイナー・サイクルの数を表す整数値がこのコンポーネントに返されます。

progs 指定したスケジューラーで1マイナー・サイクル中にスケジュールして実行することが可能なプログラムの最大数。

max 指定したスケジューラーで同時にスケジュール可能なプログラムの最大数。

configflg	指定したスケジューラーに割り当てられたアクセス権。
fbs_id	指定したFrequency-Based Schedulerの識別子を表す一意の正の整数値。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **`fbsconfigure(3)`** の man ページを参照して下さい。

fbscycle - マイナー・サイクル/メジャー・フレームの総数を返答

本ルーチンは Frequency-Based Scheduler の現在のマイナー・サイクルおよびメジャー・フレームの総数値を取得します。これらの値でシミュレーションの進捗を判定することが可能です。

概要

```
#include <fbsched.h>

int fbscycle (fbs_id, cycle_buf)
int fbs_id;
struct fbscycle_ds {
    int ccycle;
    int cframe;
} *cycle_buf;
```

パラメータ

fbs_id	Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は**fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。 識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。
cycle_buf	指定したスケジューラーの現在のマイナー・サイクルとメジャー・フレームを示す整数値を**fbscycle**が返すfbscycle_ds構造体を参照します。*ccycle*コンポーネントはサイクルの番号を含みます。*cframe*コンポーネントはフレームの番号を含みます。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために errno が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbscycle(3)** の man ページを参照して下さい。

fbsdetach - Frequency-Based Schedulerからタイミング・ソースを分離

本ルーチンは現在接続されているタイミング・ソースを Frequency-Based Scheduler から分離、またはエンド・オブ・サイクル・スケジューリングを無効にします。タイミング・ソースがリアルタイム・クロックである場合、本ルーチンを呼び出す前にそのクロックを停止することを推奨します。**fbsrunrtc** (本ルーチンの説明は 6-25 ページを参照)の呼び出しを行うことでそうすることが可能です。

概要

```
#include <fbsched.h>

int fbsdetach (fbs_id)
int fbs_id;
```

パラメータ

fbs_id — Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために errno が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbsdetach(3)** の man ページを参照して下さい。

fbsdir - スケジューラー・キーのリストを返答

本ルーチンは現在システム構成されている Frequency-Based Scheduler に関連付けられたキーのリストを返します。

概要

```
#include <fbsched.h>

int fbsdir (key_t *keylist, size_t *keycnt)
```

パラメータ

keylist	システムに構成されているFrequency-Based Schedulerのキーが返されるメモリ予約へのポインター。この領域は*keycnt*で指定された`key_t`型エレメントの数を格納するのに十分な大きさである必要があります。
keycnt	システムに構成されているFrequency-Based Schedulerのキーの数。この数がシステムに構成されているスケジューラーの数未満である場合、エラーが返されます。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **`fbsdir(3)`** の man ページを参照して下さい。

fbsgetpid - スケジュール済みプロセスのプロセスIDを返答

本ルーチンは Frequency-Based Scheduler の現在のマイナー・サイクルおよびメジャー・フレームの総数値を取得します。これらの値でシミュレーションの進捗を判定することが可能です。

概要

```
#include <fbsched.h>

int fbsgetpid (fbs_id, name_ptr, fpid)
int fbs_id, fpid;
char *name_ptr;
```

パラメータ

fbs_id — Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

name_ptr — プロセスを特定するパス名称へのポインター。本引数は現在無視されておりNULLにする必要があります。プロセス名称を指定してプロセスIDを取得するには**namepid(3)**または**nametopid(3)**を使用して下さい。

fpid — プロセスに関する一意のFrequency-Based SchedulerプロセスIDを提供する整数値。この値は**sched_pgmadd** (本ルーチンの説明は6-51ページを参照)の呼び出しをした時に取得します。

戻り値

正常に終了すると **fbsgetpid** はプロセス ID を返します。戻り値-1 はエラーが発生したことを示します(エラーを示すために errno が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbsgetpid(3)** の man ページを参照して下さい。

fbsgetrtc - リアルタイム・クロックの現在値を取得

本ルーチンは Frequency-Based Scheduler に接続されたリアルタイム・クロックに関する現在のカウントおよび分解能を取得します。

概要

```
#include <fbsched.h>

int fbsgetrtc （fbs_id, count, resolution）
int  fbs_id, *count, *resolution ;
```

パラメータ

fbs_id	Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。 識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。
count	マイナー・サイクルあたりの現在のクロック・カウント数。この値は1から65535の範囲となります。
resolution	現在の1クロック・カウントの持続時間をマイクロ秒単位で示す整数値。この値は1, 10, 100, 1000, 10000のいずれかです。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために errno が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbsgetrtc(3)** の man ページを参照して下さい。

fbsid - キーに対するFrequency-Based Scheduler識別子を返答

本ルーチンは固有のユーザー指定キーに関連付けられた Frequency-Based Scheduler 識別子を取得します。このキーは **fbsconfigure(3)** の呼び出しを介してスケジューラーが生成された時に指定したキーと一致する必要があります。

概要

```
#include <fbsched.h>

int fbsid (fbs_key)
int fbs_key;
```

パラメータ

fbs_key	Frequency-Based Schedulerを特定する整数値。この値は **fbsconfigure(3)** (本ルーチンの説明は6-10ページ参照)を使ってスケジューラーが生成された時に*key*に対して指定した値と同じです。

戻り値

正常に終了すると **fbsid** はキーに関連付けられた一意の Frequency-Based Scheduler 識別子を表す整数値を返します。戻り値-1 はエラーが発生したことを示します(エラーを示すために errno が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbsid(3)** の man ページを参照して下さい。

fbsinfo, fbsinfo_big - Frequency-Based Schedulerの情報を返答

これらのルーチンは他のルーチンの呼び出しでは取得できない Frequency-Based Scheduler に関連する情報を取得します(例えば、**sched_fbsqry**, **sched_pgmqry**)。

情報には以下が含まれます：

- スケジューラーの所有者および作成者のユーザーIDとグループID
- スケジューラーに割り当てられたアクセス権
- スケジューラーの識別子に関連付けられたキー
- スケジューラー上の全プロセスの合計オーバーラン数
- システムでアクティブなCPU
- Performance Monitorが有効になっているCPU
- FBS有効フラグ
- スケジューラーに接続されたデバイスのパス名称

これら 2 つのルーチンは CPU マスクの処理以外は同一です。詳細は 6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int fbsinfo (fbs_id, info_buf, devname)
int fbs_id;
struct fbsinfo_ds {
    int uid;
    int gid;
    int cuid;
    int cgid;
    int mode;
    int key;
    int flags;
    int devid;
    int overruns;
    int cpuactive;
    int pm_cpuactive;
    int enabled;
    int filler[29];
} *info_buf
char *devname;

int fbsinfo_big (fbs_id, info_buf, devname)
int fbs_id;
struct fbsinfo_big_ds {
    int uid;
    int gid;
    int cuid;
    int cgid;
    int mode;
    int key;
    int flags;
```

```
    int devid;
    int overruns;
    int enabled;
    int ifill[9];
    cpuset_t *cpuset_active;
    cpuset_t *pm_cpuset_active;
} *info_buf;
char *devname;
```

パラメータ

fbs_id

Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は**fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

info_buf

fbsinfoまたは**fbsinfo_big**がそれぞれ指定したスケジューラーに関する情報を返すfbsinfo_dsまたはfbsinfo_big_ds構造体。構造体の各コンポーネントに返される情報を以下に示します。

uid	所有者のユーザーID
gid	所有者のグループID
cuid	作成者のユーザーID
cgid	作成者のグループID
mode	アクセス・モード
key	キー
flags	フラグ・ワード
devid	将来使用のため予約済み
overruns	スケジューラー上の全プロセスのハード・オーバーランの総数
cpuactive	システムでアクティブなCPUマスク
pm_cpuactive	Performance Monitorが有効になっているCPUマスク
cpuset_active	システムの**cpuset(3)**書式によるCPUマスクへのポインター。これがNULLポインターである場合、cpusetcpuset情報は返されません。
pm_cpuset_active	Performance Monitorが有効になっている**cpuset(3)**書式によるCPUマスクへのポインター。これがNULLポインターである場合、cpusetcpuset情報は返されません。
enabled	FBS有効化フラグ

devname	**fbsinfo**および**fbsinfo_big**が指定したFrequency-Based Schedulerのタイミング・ソースとして使用しているデバイスのパス名称を返す変数。エンド・オブ・サイクル・スケジューリングが指定されている場合、*devname*はNULL文字列になります。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbsinfo(3)** および **fbsinfo_big(3)** の man ページを参照して下さい。

fbsintrpt - スケジューリングを開始/停止/再開

本ルーチンは Frequency-Based Scheduler のスケジューリングを開始、停止、再開します。スケジューリングを開始するために本ルーチンを呼び出した場合、マイナー・サイクル、メジャー・フレーム、オーバーラン総数値はリセットされます。スケジューリングを再開するために呼び出した場合、それらの値はリセットされません。

fbsintrpt を呼び出す前にエンド・オブ・サイクル・スケジューリングを指定、またはスケジューリングを開始する Frequency-Based Scheduler にタイミング・ソースを接続するために **fbsattach** を起動する必要があります(**fbsattach** の説明は 6-8 ページを参照)。

タイミング・ソースとしてリアルタイム・クロックを指定した場合、クロックを設定し開始するまでスケジューリングは始まりません(**fbssetrtc** と **fbsrunrtc** の説明はそれぞれ 6-28 ページと 6-25 ページを参照)。

概要

```
#include <fbsched.h>

int fbsintrpt （fbs_id, intrflag）
int fbs_id, intrflag;
```

パラメータ

fbs_id — Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

intrflag — 指定したスケジューラー上のプロセスのスケジューリングを開始、停止、または再開するのかを示す整数値。許容値および対応する結果は次のとおりとなります：

- <0　最初のフレーム、サイクル、オーバーラン総数の値をゼロに設定してスケジュール中のプロセスを開始します
- 0　スケジュール中のプロセスを停止し、現在のフレームとサイクルの総数を保存します
- >0　スケジューラーを最後に停止した時に保存された値をフレーム、サイクル、オーバーラン総数の値に設定してスケジュール中のプロセスを再開します

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために errno が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbsintrpt(3)** の man ページを参照して下さい。

fbsremove - Frequency-Based Schedulerを削除

本ルーチンは Frequency-Based Scheduler を削除してそれに関連付けられたデータ構造体を解放します。**fbsremove** を呼び出す前にタイミング・ソースがスケジューラーから分離されていることを確認する必要があります(**fbsdetach** の使用に関する情報は 6-14 ページを参照)。

fbsremove は指定したスケジューラーにスケジュールされている全てのプロセスを削除することに注意することが重要です。**fbsremove** を呼び出す前にスケジュールされている全てのプロセスを削除することを推奨します。**pgmremove** の呼び出し(本ルーチンの使用に関する情報は 6-33 ページを参照)を行う事でそうすることが可能です。

Frequency-Based Scheduler を削除するには、呼び出し元プロセスはスケジューラーの所有者/作成者の実行ユーザーID と同じ実行ユーザーID を持っている必要があることに注意して下さい。

概要

```
#include <fbsched.h>

int fbsremove (fbs_id, ab)
int fbs_id, ab;
```

パラメータ

fbs_id — Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

ab — スケジューラーにスケジュールされているプロセスを処理する方法を示す整数値。許容値および対応する結果は次のとおり：

- <0　指定したスケジューラーに現在スケジュールされている全てのプロセスを終了して削除します
- ≧0　指定したスケジューラーに現在スケジュールされている全てのプロセスを削除します

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために errno が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbsremove(3)** の man ページを参照して下さい。

fbsresume - Frequency-Based Schedulerのスケジュールを再開

fbsresume ライブラリ・ルーチンは指定したマイナー・サイクル、メジャー・フレーム、オーバーラン総数で Frequency-Based Scheduler のプロセスのスケジューリングを再開します。

Frequency-Based Scheduler のプロセスのスケジューリングを再開するには、呼び出し元プロセスがスケジューラへの変更許可を持っている必要があります。

スケジューラーの現在のフレーム、サイクル、オーバーランの値を変更せずに Frequency-Based Scheduler のプロセスのスケジューリングを再開したい場合、**fbsintrpt(3)** ルーチン(本ルーチンの説明は 6-22 ページを参照)を使用することを推奨します。

CAUTION

fbsresume ルーチンは指定したスケジューラーにスケジュールされている全てのプロセスに関する Performance Monitor の値を消去します。スケジューラーのフレームおよびサイクルの総数を変更すると Performance Monitor で保持している値が不正確となります。

概要

```
#include <fbsched.h>

int fbsresume (fbs_id, frame, cycle, overruns)
int fbs_id, frame, cycle, overruns;
```

パラメータ

fbs_id — Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされている Frequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

frame — 指定したスケジューラーでプロセスのスケジューリングを再開したいメジャー・フレームを示す整数値。

cycle — 指定したスケジューラーでプロセスのスケジューリングを再開したいマイナー・サイクルを示す整数値。

この値は0からフレームあたりのマイナー・サイクルの総数-1の範囲とすることが可能です。フレームあたりのマイナー・サイクルの総数は、**fbsconfigure** (6-10ページ参照)を呼び出しを行ってスケジューラーが生成された時に指定しています。

overruns — 指定したスケジューラーでスケジューリングを再開する時にオーバーラン総数に設定したい値を示す整数値。

オーバーラン総数を変更したくない場合、値は-1を指定して下さい。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbsresume(3)** の man ページを参照して下さい。

fbsrunrtc - リアルタイム・クロックを開始/停止

本ルーチンは Frequency-Based Scheduler に接続されているリアルタイム・クロックのカウントを開始または停止します。

概要

```
#include <fbsched.h>

int fbsrunrtc （fbs_id, runflag）
int fbs_id, runflag ;
```

パラメータ

fbs_id — Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

runflag — リアルタイム・クロックを開始するのか停止するのかを示す整数値。ゼロ以外の値はクロックを開始することを示します。値ゼロはクロックを停止することを示します。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbsrunrtc(3)** の man ページを参照して下さい。

fbsschedself - Frequency-Based Schedulerに呼び出し元プロセスを追加

本ルーチンは Frequency-Based Scheduler に呼び出し元プロセスをスケジュールします。

fbsschedself はプロセスにそのスケジューリング・ポリシーや優先度またはその CPU バイアスの設定を許可していないことに注意することが重要です。これらの作業は **fbsschedself** を呼び出す前に実行する必要があります。

Frequency-Based Scheduler に**/idle** を追加するために本ルーチンを使用できないことに注意して下さい。

Frequency-Based Scheduler に呼び出し元プロセスをスケジュールするには、呼び出し元プロセスはスケジューラーに対する変更許可を持っている必要があります。

FBS でスケジュールされたプロセスのスケジューリング・ポリシーや優先度を変更するには、**sched_pgmresched** ルーチン(本ルーチンの説明は 6-65 ページを参照)を使用して下さい。

概要

```
#include <fbsched.h>

int fbsschedself (fbs_id, name, sched_buf)
int fbs_id;
char *name;
struct fbssched_buf {
    int version;
    int param;
    int period;
    int cycle;
    int ab;
    int fpid;
} *sched_buf;
```

パラメータ

fbs_id Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は**fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

name 呼び出し元プロセスに関連付けられたプログラムを特定する標準パス名称または任意の情報。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。

sched_buf スケジュールされるプロセスに関するスケジューリング情報を含むfbssched_buf構造体。各コンポーネントで指定される情報の種類は以下のとおり。

- *version* **fbsschedself**に渡される*sched_buf*のバージョンを示す整数値。この目的用に**<fbsched.h>**に定義されている記号定数**FBSSCHED_BUF_V1**を指定して下さい。
- *param* FBSでスケジュールされるプロセスに渡される整数値。この値は**sched_pgmqry** (本ルーチンの説明は6-61ページを参照)の呼び出しを通してFBSにスケジュールされたプロセスにより取得することが可能です。

period 呼び出し元プロセスが各メジャー・フレームで起動される頻度を示す整数値。頻度1は呼び出し元プロセスがマイナー・サイクル毎に起動されることを示し、頻度2はマイナー・サイクル2回毎に1回起動されることを示します。

この値は1から**fbsconfigure** (本ルーチンの説明は6-10ページを参照)の呼び出しで定義した時に指定したスケジューラーのフレームを構成するマイナー・サイクルの数の範囲とすることが可能です。

cycle 呼び出し元プロセスが各フレームで起動するようにスケジュールされている最初のマイナー・サイクルを示す整数値。この値は0からフレームあたりのマイナー・サイクルの総数-1の範囲とすることが可能です。フレームあたりのマイナー・サイクルの総数は、**fbsconfigure** (本ルーチンの説明は6-10ページを参照)の呼び出しで指定されています。

ab 呼び出し元プロセスでフレーム・オーバーランが発生した場合にスケジューラーを停止させるべきかどうかを示す整数値。ゼロ以外の値はスケジューラーを停止することを示します。

fpid **fbsschedself**が返す整数値で、これはスケジュールされるプロセスに関する一意のFrequency-Based SchedulerプロセスIDです。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbsschedself(3)** の man ページを参照して下さい。

fbssetrtc - リアルタイム・クロックを設定

本ルーチンは FBS のタイミング・ソースとして使用されるリアルタイム・クロックのカウントと分解能の値を設定することでマイナー・サイクルの持続時間を規定します。

概要

```
#include <fbsched.h>

int fbssetrtc （fbs_id, count, resolution）
int fbs_id, count, resolution ;
```

パラメータ

fbs_id — Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は**fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

count — マイナー・サイクルあたりのクロック・カウントの数を示す整数値。この値は2から65535の範囲が可能です。

resolution — 1クロック・カウントのマイクロ秒単位での持続期間を示す整数値。この値は次のいずれかにする必要があります：1, 10, 100, 1000, 10000。

注意事項： *count* に 1 を使用することは出来ませんが、*resolution* の値と等しいタイミング間隔は、次に低い *resolution* の値と *count* に 10 を指定することにより設定することが可能となります。例えば、*resolution* が 1000 および *count* が 10 で 10,000 マイクロ秒のタイミング間隔をもたらします。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbssetrtc(3)** の man ページを参照して下さい。

fbstrig - スリープ中のFrequency-Based Schedulerプロセスを実行可能に遷移

本ルーチンはFrequency-Based Schedulerにスケジュールされているプロセスが**fbswait(3)**の中でスリープ状態である場合に実行可能にします。更にプロセスが実行中のプロセッサーにコンテキスト・スイッチを強制することが可能です。

概要

```
#include <fbsched.h>

int fbstrig (fpid, tgrflg)
int fpid, tgrflg;
```

パラメータ

fpid	スリープ中のプロセスに関する一意のFrequency-Based Schedulerプロセスを提供する整数値。この値は**sched_pgmadd** (本ルーチンの説明は6-51ページを参照)の呼び出しをした時に取得します。
tgrflg	起動されるプロセスが実行中のプロセッサーにコンテキスト・スイッチを強制するのかどうかを示す整数値。ゼロ以外の値はコンテキスト・スイッチを強制することを示します。

戻り値

戻り値0は呼び出しが成功したことを示します。戻り値-1はエラーが発生したことを示します(エラーを示すために errno が設定されます)。発生する可能性のあるエラーの種類を表示するには**fbstrig(3)**のmanページを参照して下さい。

fbswait - Frequency-Based Schedulerを待機

本ルーチンは Frequency-Based Scheduler にスケジュールされているプロセスを次にスケジュールされているマイナー・サイクルまでスリープすることを可能にします。

スケジュールされているプロセスが次にスケジュールされているマイナー・サイクルまでに本サービスをコールしない場合、ソフト・オーバーランまたはハード・オーバーランのどちらかが発生します。

スケジュールされているマイナー・サイクルを連続的に逃したプロセス毎の総数がプロセス毎のソフト・オーバーラン上限に達していないまたは超えていない場合、ソフト・オーバーランが発生します。ソフト・オーバーランが発生した時、次にスケジュールされているマイナー・サイクルを待つためのブロックの代わりにプロセスは **fbswait** 呼び出しからすぐに戻ります。

スケジュールされているマイナー・サイクルを連続的に逃した総数がプロセス毎のソフト・オーバーラン上限に達したまたは超えた場合、ハード・オーバーランが発生します。この場合、プロセスは次にスケジュールされているマイナー・サイクルまで **fbswait** でブロックします。

プロセスの連続的なソフト・オーバーラン上限は、**sched_pgm_set_soft_overrun_limit(3)**を使って既定値の 0 から変更することが可能です。**pmqrypgm(3)**を使って読み取り可能なハード・オーバーラン総数および **sched_pgm_soft_overrun_query(3)**を使って読み取り可能なソフト・オーバーラン総数はプロセスが割り当てられた頻度で実際に動作しているかどうかを示します。

スケジュールされているプロセスがデッドラインの影響下にあり、スケジュールされているプロセスがそのデッドライン時間を過ぎた後にこのサービスを呼び出す場合、デッドライン違反を検出してスケジューラーは停止される可能性があります。

概要

```
#include <fbsched.h>

int fbswait (void)
```

戻り値

戻り値 0 はプロセスが Frequency-Based Scheduler により起動されたことを示します。戻り値 1 はプロセスが **fbstrig(3)**により起動されたことを示します。戻り値 2 はカーネルがソフト・オーバーランを検出し、プロセスにそこからの回復を試みることを許可するためにスリープしなかったことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために errno が設定されます)。発生する可能性のあるエラーの種類を表示するには **fbswait(3)**の man ページを参照して下さい。

nametopid, namepid, nametopid_big, namepid_big - 指定したプロセス名称のプロセスIDを返答

namepid と **namepid_big** ルーチンは指定したプロセスのプロセス ID (pid)を返します。**nametopid** と **nametopid_big** ルーチンは同じ機能を実行しますが、検索を特定の Frequency-Based Scheduler にスケジュールされているプロセスに限定することが可能です。

「big」ルーチンは CPU マスクの処理を除けば非 big ルーチンと全く同じです。違いの詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int nametopid (name, fbs_key, cpu)
int namepid (name, cpu)
int fbs_key, cpu;
char *name;

int nametopid_big (name, fbs_key, cpuset)
int namepid_big (name, cpuset)
int fbs_key;
char *name;
cpuset_t *cpuset;
```

パラメータ

name プロセスを特定するパス名称

fbs_key Frequency-Based Schedulerを特定する整数値。この値は **fbsconfigure(3)** (本ルーチンの説明は6-10ページを参照)を使ってスケジューラーが生成された時に*key*に対して指定した値と同じです。

cpu プロセスを特定するために*name*パラメータの値と関連して使用されるプロセッサーを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
-1	いずれかのプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
bit mask	(*cpu* & (1 << *i*))で設定したプロセッサーが要求されます(*i*は0から15の範囲の整数でCPUを表します)

cpuset pidが返されるプロセスを特定するため、*name*と関連して使用されるプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

`FBS_CPUSET_CURCPU`
: 呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセス

`FBS_CPUSET_ALLCPUS`
: 全てのプロセッサーで実行中の*name*で指定された最初のプロセス

`FBS_CPUSET_ONECPU(`*cpu*`)`
: CPU IDが*cpu*のプロセッサーで実行中の*name*で指定された最初のプロセス

*cpuset*が空の**`cpuset(3)`**を指し示している場合、その動作は`FBS_CPUSET_CURCPU`で説明しているとおりです。

戻り値

成功するとこれらの関数はプロセス ID を返します。失敗するとこれらの関数は-1 を返し、エラーを示すために `errno` が設定されます。発生する可能性のあるエラーの種類を表示するには **`nametopid(3)`** と **`nametopid_big(3)`** の man ページを参照して下さい。

pgmremove, pgmremove_big - Frequency-Based Schedulerからプロセスを削除

これらのルーチンは Frequency-Based Scheduler からプロセスを削除します。以下の方法のいずれかを使って削除するプロセスを特定して下さい：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based SchedulerプロセスIDを指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based SchedulerプロセスIDを指定

NOTE

同じ CPU 上に複数回スケジュールされたプロセスを特定することが可能な唯一の方法は、その Frequency-Based Scheduler プロセス ID を指定することです。

これらの 2 つのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int pgmremove （fbs_id, name, cpu, fpid, ab）
int fbs_id, cpu, fpid, ab;
char *name;

int pgmremove_big （fbs_id, name, cpuset, fpid, ab）
int fbs_id, fpid, ab;
char *name;
cpuset_t *cpuset;
```

パラメータ

fbs_id — Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

name — 指定したスケジューラーから削除するプロセスを特定する標準パス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。本変数がNULL文字列である場合、*fpid*パラメータにFrequency-Based SchedulerプロセスIDを提供する必要があります。

cpu — 指定したプロセッサーから削除するプロセスを特定するために*name*パラメータの値と関連して使用されるプロセッサを示す整数値。許容値および対応する結果は次のとおり：

	0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが削除されます
	-1	全てのプロセッサーで現在実行中の*name*で指定された最初のプロセスが削除されます
	bit mask	(*cpu* & (1 << *i*))が設定され(*i*は0から15の範囲の整数でCPUを表します)、それが唯一のビット・セットである場合、CPU *i*で実行中の*name*で指定された最初のプロセスが削除されます
		(*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name*で指定された最初のプロセスが削除されます

cpuset

問合せするプロセスを特定するため、*name*と関連して使用されるプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

`FBS_CPUSET_CURCPU`
: *name*で指定され、呼び出しが行われたプロセッサーで現在実行中の最初に見つかったプロセス

`FBS_CPUSET_ALLCPUS`
: いずれかのプロセッサーで実行中の*name*で指定され最初に見つかったプロセス

`FBS_CPUSET_ONECPU(`*cpu*`)`
: *name*で指定され、CPU IDが*cpu*のプロセッサーで実行中の最初に見つかったプロセス

*cpuset*が空の**cpuset(3)**を指し示している場合、その動作は`FBS_CPUSET_CURCPU`で説明しているとおりです。

fpid

削除するプロセスに関する一意のFrequency-Based Schedulerプロセス IDを提供する整数値。この値は**sched_pgmadd** (本ルーチンの説明は6-51ページを参照)の呼び出しを行ったときに取得します。*name*と*cpu*を指定するだけで削除するプログラムを特定することを選択する場合、この値は-1にする必要があります。

ab

指定したスケジューラーから指定したプロセスを削除する方法を示す整数値を含むフラグ。正の値はスケジューラーからプロセスは削除されますが実行を継続することを示します。負の値はプロセスはスケジューラーから削除され終了することを示します。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **pgmremove(3)** および **pgmremove_big(3)** の man ページを参照して下さい。

pgmtrigger - Frequency-Based Schedulerのプロセスを起動

本ルーチンは **fbswait** でスリープ状態にあるプロセスを起こす処理を可能にします。呼び出し元プロセスは Frequency-Based Scheduler にスケジュールされている必要はなく、ターゲットのプロセスはそうである必要があることに注意することが重要です。

概要

```
#include <fbsched.h>

int pgmtrigger （fpid, tgrflg）
int fpid, tgrflg;
```

パラメータ

fpid	スリープ中のプロセスに関する一意のFrequency-Based Schedulerプロセス IDを提供する整数値。この値は**sched_pgmadd** (本ルーチンの説明は6-51ページを参照)の呼び出しを行ったときに取得します。
tgrflg	起動されたプロセスが実行しているプロセッサーでコンテキスト・スイッチを強制するかどうかを示す整数値。ゼロ以外の値はコンテキスト・スイッチが強制されることを示します。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します (エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **pgmtrigger(3)** の man ページを参照して下さい。

sched_fbsqry, sched_fbsqry_big - Frequency-Based Schedulerのプロセスを問合せ

これらのルーチンは Frequency-Based Scheduler にスケジュールされたプロセスに関する情報を取得します。情報はユーザー指定のプロセッサー上にスケジュールされた全てのプロセスについて返されます。各プロセスについて提供される情報は以下が含まれます：

- プロセスが実行可能なCPUのマスク
- Frequency-Based SchedulerプロセスID
- スケジュールされているプロセスのポリシー
- スケジューリングの優先度
- 周期(プロセスが各メジャー・フレームで起動される頻度を示すマイナー・サイクルの数)
- 開始ベース・サイクル(プロセスが各メジャー・フレームで起動される予定の最初のマイナー・サイクル)
- 「halt on overrun」フラグの値
- 現在のプロセスの状態

これらの 2 つのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int sched_fbsqry (fbs_id, cpu, fps_buf, buf_cnt)
int fbs_id, cpu, buf_cnt;
struct pgm2_ds {
    char *name_ptr;
    int cpu;
    int fpid;
    int cid;
    int prior;
    int param;
    int period;
    int cycle;
    int halt;
    int status;
} *fps_buf;

int sched_fbsqry_big (fbs_id, cpuset, fps_buf, buf_cnt)
int fbs_id, buf_cnt;
struct pgm2_big_ds {
    char *name_ptr;
    cpuset_t *cpuset;
    void *vfill[9];
    int fpid;
    int cid;
    int prior;
    int param;
    int period;
    int cycle;
```

```
    int halt;
    int status;
    int ifill[9];
} *fps_buf;
cpuset_t *cpuset;
```

パラメータ

fbs_id
: Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は**fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

  識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

cpu
: スケジューリング情報を取得するプロセッサを示す整数値。許容値および対応する結果は次のとおり：

  0
  : 呼び出しが行われたプロセッサーで実行中のプロセスに関するスケジューリング情報を返します

  -1
  : スケジューラー上の全てのプロセスに関するスケジューリング情報を返します

  bit mask
  : (*cpu* & (1 << *i*))が設定された場合(*i*は0から15の範囲の整数でCPUを表します)、CPU *i*で実行中のプロセスに関するスケジューリング情報を返します

cpuset
: 情報を取得するFBSでスケジュールされたプロセスのセットを特定するため、*name_ptr*と関連して使用されるプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

  FBS_CPUSET_CURCPU
  : *name_ptr*で指定され、呼び出しが行われたプロセッサーで現在実行中の見つかったプロセスのセット

  FBS_CPUSET_ALLCPUS
  : いずれかのプロセッサーで実行中の*name_ptr*で指定され見つかったプロセスのセット

  FBS_CPUSET_ONECPU(*cpu*)
  : *name_ptr*で指定され、CPU IDが*cpu*のプロセッサーで実行中の見つかったプロセスのセット

  *cpuset*が空の**cpuset(3)**を指し示している場合、その動作はFBS_CPUSET_CURCPUで説明しているとおりです。

fbs_buf
: *cpu*または*cpuset*パラメータで指定されたプロセッサー上の各プロセスに関するスケジューリング情報を**sched_fbsqry**または**sched_fbsqry_big**がそれぞれ返すpgm2_dsまたはpgm2_big_ds構造体へのポインター。単一プロセスに関する構造体の各コンポーネントに返される情報の種類は次のとおりです：

name_ptr	情報が返されるプロセスを特定するパス名称を含む変数へのポインター。アプリケーション**free**は(**malloc(3)**を参照)を使って全ての有効なエントリーに対して*name_ptr*に関連付けられたメモリの割り当てを解放する必要があることに注意して下さい。
cpu	プロセスが実行可能なプロセッサーを示すビット・マスク
cpuset	プロセスが実行可能なプロセッサーのセットが返される*cpuset*へのポインター。これがNULLポインターである場合、プロセスに関する*cpuset*の情報は返されません。
fpid	プロセスのFrequency-Based SchedulerプロセスID
cid	プロセスのスケジューリング・ポリシー：SCHED_FIFO, SCHED_RR, SCHED_OTHER
prior	指定したプロセスのスケジューリング優先度を示す整数値
param	プロセスの開始パラメータ(オプション)
period	プロセスが各メジャー・フレームで起動される頻度を示すマイナー・サイクルの数
cycle	プロセスが各メジャー・フレームで起動される予定の最初のマイナー・サイクル(開始ベース・サイクル)
halt	「halt on overrun」フラグの値。ゼロ以外の値はフラグが設定されることを示します。値が0はフラグが設定されないことを示します。
status	**<fbsched.h>**に定義されているプロセスの現在の状況

buf_cnt *fbs_buf*が示す配列内の構造体の数を示す整数値。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **sched_fbsqry(3)** または **sched_fbsqry_big(3)** の man ページを参照して下さい。

sched_pgm_deadline_query, sched_pgm_deadline_query_big - プロセスに割り当て済みデッドライン時間を問合せ

これらのルーチンは Frequency-Based Scheduler に現在スケジュールされているプロセスに関するデッドラインのパラメータを問い合わせます。

この機能は新しいデッドライン違反を検出しないことに注意することが重要です。新しいデッドライン違反の検出を実行するには **sched_pgm_deadline_test(3)** を使用して下さい。

これらの 2 つのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int sched_pgm_deadline_query (fbs_id, deadline_info_buf)
int fbs_id;
struct deadline_info_ds {
    char *name_ptr;
    int cpu;
    int fpid;
    int halt;
    deadline_kind kind;
    deadline_origin origin;
    struct timespec deadline;
    int total_violations;
} *deadline_info_buf;

int sched_pgm_deadline_query_big (fbs_id, deadline_info_buf)
int fbs_id;
struct deadline_info_big_ds {
    char *name_ptr;
    cpuset_t cpuset;
    void *vfill[9];
    int fpid;
    int halt;
    int ifill[9];
    deadline_kind kind;
    deadline_origin origin;
    struct timespec deadline;
    int total_violations;
} *deadline_info_buf;
```

パラメータ

fbs_id — Frequency-Based Schedulerの識別子を表す一意の正の整数値。この値は**fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

deadline_info_buf — プロセスに割り当てられたいずれのデッドラインの属性を含むdeadline_info_dsまたはdeadline_info_big_ds構造体へのポインター。構造体の定義は次のとおりです：

name_ptr プロセスを特定するパス名称へのポインター。パス名称がNULL文字列である場合、*fpid*領域は指定されている必要があります。

cpu 指定したプログラムの実行をスケジュール可能なプロセッサーを示す整数値。許容値および対応する結果は次のとおり：

- 0 — *name_ptr*で指定されたプログラムは呼び出しが行われるプロセッサーにスケジュールすることが可能です
- -1 — *name_ptr*で指定されたプログラムはどのプロセッサーでもスケジュールすることが可能です
- bit mask — (*cpu* & (1 << *i*))が設定された場合(*i*は0から15の範囲の整数でCPUを表します)、*name_ptr*で指定されたプログラムはCPU *i* にスケジュールすることが可能です。

cpuset 問い合わせるプロセスを特定するため、*name_ptr*と関連して使用されるプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

- FBS_CPUSET_CURCPU — *name_ptr*で指定され、呼び出しが行われたプロセッサーで現在実行中の最初に見つかったプロセス
- FBS_CPUSET_ALLCPUS — いずれかのプロセッサーで実行中の*name_ptr*で指定され最初に見つかったプロセス
- FBS_CPUSET_ONECPU(*cpu*) — *name_ptr*で指定され、CPU IDが*cpu*のプロセッサーで実行中の最初に見つかったプロセス

*cpuset*が空の**cpuset(3)**を指し示している場合、その動作はFBS_CPUSET_CURCPUで説明しているとおりです。

fpid 一意のFrequency-Based SchedulerプロセスIDを提供する整数値。この値が-1である場合、*name_ptr*が提供される必要があります。

halt プロセスのデッドライン違反の検出でスケジューラーを停止するかどうかを示します。この値はスケジューラーを停止することを示すDL_HALT、もしくはDL_NOHALTになります。

kind **sched_pgm_set_deadline(3)**で設定されたデッドラインの種類。

この値は*deadline*の解釈に影響を及ぼします。*kind*はデッドラインを指定したプロセスに適用しないことを示すDEADLINE_CLEAR、もしくはCLOCK_MONOTONICデッドライン時間を指定したプロセスに割り当てることを示すDEADLINE_WALL_TIMEのいずれかとなります。

origin デッドライン時間が測定されるポイント。*origin*はスケジュールされたタスクのサイクルの開始からデッドライン時間が測定されることを示すDL_CYCLE_RELATIVE、もしくはスケジュールされたプロセスが**fbswait(3)**を終了して実行を開始する時間からデッドライン時間が測定されることを示すDL_TASK_RELATIVEのいずれかとなります。

deadline **fbswait(3)**に戻る前にプロセスが実行に費やすことが期待される最大時間。

total_violations 指定したプロセスまたはタスクで発生したデッドライン違反の総数。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **sched_pgm_deadline_query(3)**または **sched_pgm_deadline_query_big(3)** の man ページを参照して下さい。

sched_pgm_deadline_test, sched_pgm_deadline_test_big - デッドライン違反の有無をテスト

これらのルーチンは Frequency-Based Scheduler にスケジュールされたプロセスによるデッドライン違反の発生についてテストします。

これらの 2 つのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int sched_pgm_deadline_test (fbs_id, violation_buf)
int fbs_id;
struct violation_ds {
    char *name_ptr;
    int cpu;
    int fpid;
    int violated;
    int total_violations;
    deadline_kind kind;
    deadline_origin origin;
    struct timespec remaining;
} *violation_buf;

int sched_pgm_deadline_test_big (fbs_id, violation_buf)
int fbs_id;
struct violation_big_ds {
    char *name_ptr;
    cpuset_t cpuset;
    void *vfill[9];
    int fpid;
    int violated;
    int total_violations;
    int ifill[9];
    deadline_kind kind;
    deadline_origin origin;
    struct timespec deadline;
} *violation_buf;
```

パラメータ

fbs_id — Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

violation_buf — スケジュールされたプロセスに関するデッドライン違反状態の情報を含むdeadline_info_bigまたはdeadline_info_big_ds構造体へのポインター。構造体の定義は次のとおりです：

name_ptr プロセスを特定するパス名称へのポインター。パス名称がNULL文字列である場合、*fpid*領域は指定されている必要があります。

cpu 指定したプロセスの実行をスケジュール可能なプロセッサーを示す整数値。許容値および対応する結果は次のとおり：

- 0 — *name_ptr*で指定されたプログラムは呼び出しが行われるプロセッサーにスケジュールすることが可能です
- -1 — *name_ptr*で指定されたプログラムはどのプロセッサーでもスケジュールすることが可能です
- bit mask — (*cpu* & (1 << *i*))が設定された場合(*i*は0から15の範囲の整数でCPUを表します)、*name_ptr*で指定されたプログラムはCPU *i* にスケジュールすることが可能です。

cpuset テストするプロセスを特定するため、*name_ptr*と関連して使用されるプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

- `FBS_CPUSET_CURCPU` — 呼び出しが行われたプロセッサーで現在実行中の*name_ptr*で指定された最初のプロセス
- `FBS_CPUSET_ALLCPUS` — 全てのプロセッサーで実行中の*name_ptr*で指定された最初のプロセス
- `FBS_CPUSET_ONECPU`(*cpu*) — CPU IDが*cpu*のプロセッサーで実行中の*name_ptr*で指定された最初のプロセス

*cpuset*が空の**cpuset(3)**を指し示している場合、その動作は`FBS_CPUSET_CURCPU`で説明しているとおりです。

fpid 一意のFrequency-Based SchedulerプロセスIDを提供する整数値。この値が-1である場合、*name_ptr*が提供される必要があります。

violated プロセスのデッドライン違反の検出でスケジューラーを停止するかどうかを指示して下さい。この値はスケジューラーを停止することを示すDEADLINE_VIOLATION、もしくはNO_VIOLATIONになります。

total_violations スケジューラーが開始されてからプロセスで検出されるデッドライン違反の数を指定して下さい。

kind **sched_pgm_set_deadline(3)**で設定されたデッドラインの種類。

この値は*remaining*の解釈に影響を及ぼします。*kind*はデッドラインを指定したプロセスに適用しないことを示すDEADLINE_CLEAR、もしくはCLOCK_MONOTONICデッドライン時間を指定したプロセスに割り当てることを示すDEADLINE_WALL_TIMEのいずれかとなります。

origin **sched_pgm_set_deadline(3)**でプロセスに設定されたデッドライン起点。この値は*remaining*の解釈に影響を及ぼします。*origin*はタスク実行の開始から残り時間が測定されることを示すDL_TASK_RELATIVE、サイクルの開始から残り時間が測定されることを示すDL_CYCLE_RELATIVEのいずれかとなります。

remaining デッドラインの期限までの残り時間。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **sched_pgm_deadline_test(3)**または **sched_pgm_deadline_test_big(3)**の man ページを参照して下さい。

sched_pgm_set_deadline, sched_pgm_set_deadline_big - プロセス・デッドライン時間を設定または消去

これらのルーチンは Frequency-Based Scheduler に現在スケジュールされているプロセスのデッドライン時間を設定または消去します。

これらの 2 つのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

プロセスのデッドライン時間は **fbswait(3)**に戻る前にプロセスが実行に費やすことが期待される最大時間を示します。デッドライン時間を超えた場合、デッドライン違反がプロセスで生じます。スケジューラーをオプションでデッドライン違反の検出で停止させることが可能です。

デッドラインを設定または消去するには、呼び出し元プロセスはスケジューラーに対する変更許可を持っている必要があります。

以下のいずれかの方法を使ってプロセスを特定して下さい：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based Schedulerプロセス IDを指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based Schedulerプロセス IDを指定

概要

```
#include <fbsched.h>

int sched_pgm_set_deadline (fbs_id, deadline_buf)
int fbs_id;
struct deadline_ds {
    char *name_ptr;
    int cpu;
    int fpid;
    int halt;
    deadline_kind kind;
    deadline_origin origin;
    struct timespec deadline;
} *deadline_buf;

int sched_pgm_set_deadline_big (fbs_id, deadline_buf)
int fbs_id;
struct deadline_big_ds {
    char *name_ptr;
    cpuset_t cpuset;
    void *vfill[9];
    int fpid;
    int halt;
    int ifill[9];
    deadline_kind kind;
    deadline_origin origin;
    struct timespec deadline;
} *deadline_info_buf;
```

パラメータ

fbs_id
Frequency-Based Schedulerの識別子を表す一意の正の整数値。この値は**fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

deadline_buf
スケジュールされたプロセスに関するデッドライン構成情報を含むdeadline_dsまたはdeadline_big_ds構造体へのポインター。構造体の定義は次のとおりです：

name_ptr
プロセスを特定するパス名称へのポインター。パス名称がNULL文字列である場合、*fpid*領域は指定されている必要があります。

cpu
指定したプログラムの実行をスケジュール可能なプロセッサーを示す整数値。許容値および対応する結果は次のとおり：

0
*name_ptr*で指定されたプログラムは呼び出しが行われるプロセッサーにスケジュールすることが可能です

-1
*name_ptr*で指定されたプログラムはどのプロセッサーでもスケジュールすることが可能です

bit mask
(*cpu* & (1 << *i*))が設定された場合(*i*は0から15の範囲の整数でCPUを表します)、*name_ptr*で指定されたプログラムはCPU *i* にスケジュールすることが可能です。

cpuset
デッドライン属性を変更するプロセスを特定するため、*name_ptr*と関連して使用されるプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

FBS_CPUSET_CURCPU
呼び出しが行われたプロセッサーで現在実行中の*name_ptr*で指定された最初のプロセス

FBS_CPUSET_ALLCPUS
全てのプロセッサーで実行中の*name_ptr*で指定された最初のプロセス

FBS_CPUSET_ONECPU(*cpu*)
CPU IDが*cpu*のプロセッサーで実行中の*name_ptr*で指定された最初のプロセス

*cpuset*が空の**cpuset(3)**を指し示している場合、その動作はFBS_CPUSET_CURCPUで説明しているとおりです。

fpid	一意のFrequency-Based SchedulerプロセスIDを提供する整数値。この値が-1である場合、*name_ptr*が提供される必要があります。
halt	プロセスのデッドライン違反の検出でスケジューラーを停止するかどうかを示します。この値はスケジューラーを停止することを示すDL_HALT、もしくはDL_NOHALTになります。
kind	設定されるデッドラインの種類で、*deadline*の解釈に影響を及ぼします：デッドラインを指定したプロセスに適用しないことを示すDEADLINE_CLEAR、もしくはCLOCK_MONOTONICデッドライン時間を指定したプロセスに割り当てることを示すDEADLINE_WALL_TIME。
origin	デッドライン時間が測定されるポイント。*origin*はスケジュールされたタスクのサイクルの開始からデッドライン時間が測定されることを示すDL_CYCLE_RELATIVE、もしくはスケジュールされたタスクが**fbswait(3)**を終了して実行を開始する時間からデッドライン時間が測定されることを示すDL_TASK_RELATIVEのいずれかとなります。
deadline	負ではなくINT_MAXマイクロ秒未満の値を示す必要があります。デフォルトでこの値は0秒、0ナノ秒です。すなわち、プロセスがデッドライン時間を全く設定しない場合、これはゼロです。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **`sched_pgm_set_deadline(3)`** または **`sched_pgm_set_deadline_big(3)`** の man ページを参照して下さい。

sched_pgm_set_soft_overrun_limit, sched_pgm_set_soft_overrun_limit_big - ソフト・オーバーラン上限を設定

これらのルーチンは Frequency-Based Scheduler に現在スケジュールされているプロセスの連続的なソフト・オーバーラン上限を設定します。連続的なソフト・オーバーラン上限を設定するには、呼び出し元プロセスはスケジューラーに対する変更許可を持っている必要があります。

以下のいずれかの方法を使ってプロセスを特定して下さい：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based SchedulerプロセスIDを指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based SchedulerプロセスIDを指定

これらのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int sched_pgm_set_soft_overrun_limit (fbs_id, soft_overrun_buf)
int fbs_id;
struct soft_overrun_ds {
    char *name_ptr;
    int cpu;
    int fpid;
    int soft_limit;
} *soft_overrun_buf;

int sched_pgm_set_soft_overrun_limit_big (fbs_id, soft_overrun_buf)
int fbs_id;
struct soft_overrun_big_ds {
    char *name_ptr;
    cpuset_t cpuset;
    void *vfill[9];
    int fpid;
    int soft_limit;
    int ifill[9];
} *soft_overrun_buf;
```

パラメータ

fbs_id
: Frequency-Based Schedulerの識別子を表す一意の正の整数値。この値は**fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

  識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

soft_overrun_buf
: スケジュールされたプロセスに関するソフト・オーバーラン状況を含むsoft_overrun_dsまたはsoft_overrun_big_ds構造体へのポインター。

構造体の定義は次のとおりです：

name_ptr　プロセスを特定するパス名称へのポインター。パス名称がNULL文字列である場合、*fpid*領域は指定されている必要があります。

cpu　指定したプロセスの実行をスケジュール可能なプロセッサーを示す整数値。許容値および対応する結果は次のとおり：

- 0　*name_ptr*で指定されたプログラムは呼び出しが行われるプロセッサーにスケジュールすることが可能です
- -1　*name_ptr*で指定されたプログラムはどのプロセッサーでもスケジュールすることが可能です
- bit mask　(*cpu* & (1 << *i*))が設定された場合(*i*は0から15の範囲の整数でCPUを表します)、*name_ptr*で指定されたプログラムはCPU *i* にスケジュールすることが可能です。

cpuset　オーバーラン属性を変更するプロセスを特定するため、*name_ptr*と関連して使用されるプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

- FBS_CPUSET_CURCPU　呼び出しが行われたプロセッサーで現在実行中の*name_ptr*で指定された最初のプロセス
- FBS_CPUSET_ALLCPUS　全てのプロセッサーで実行中の*name_ptr*で指定された最初のプロセス
- FBS_CPUSET_ONECPU(*cpu*)　CPU IDが*cpu*のプロセッサーで実行中の*name_ptr*で指定された最初のプロセス

*cpuset*が空の**cpuset(3)**を指し示している場合、その動作はFBS_CPUSET_CURCPUで説明しているとおりです。

fpid　一意のFrequency-Based SchedulerプロセスIDを提供する整数値。この値が-1である場合、*name_ptr*が提供される必要があります。

soft_limit　機能停止前に発生が許容される連続的なソフト・オーバーランの数。この値は負ではなくINT_MAX(**/usr/include/limits.h**を参照)未満である必要があります。デフォルトでこの値は0です。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します (エラーを示すために `errno` が設定されます)。

発生する可能性のあるエラーの種類を表示するには **`sched_pgm_set_soft_overrun_limit(3)`** または **`sched_pgm_set_soft_overrun_limit_big(3)`** の man ページを参照して下さい。

sched_pgmadd, sched_pgmadd_big - Frequency-Based Schedulerにプロセスをスケジュール

これらのルーチンは新しいプロセスを生成して Frequency-Based Scheduler にそれをスケジュールします。新しいプロセスに引数を提供するには、6-54 ページの **sched_pgmadd_args** および **sched_pgmadd_args_big** を参照して下さい。新しいプロセスに引数とソフト・オーバーランおよび/またはデッドライン属性を提供するには、6-58 ページの **sched_pgmadd_attr** および **sched_pgmadd_attr_big** を参照して下さい。

これらのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int sched_pgmadd (fbs_id, sched_buf)
int fbs_id;
struct pgm2_ds {
    char *name_ptr;
    int cpu;
    int fpid;
    int cid;
    int prior;
    int param;
    int period;
    int cycle;
    int halt;
    int status;
} *sched_buf;

int sched_pgmadd_big (fbs_id, sched_buf)
int fbs_id;
struct pgm2_big_ds {
    char *name_ptr;
    cpuset_t cpuset;
    void *vfill[9];
    int fpid;
    int cid;
    int prior;
    int param;
    int period;
    int cycle;
    int halt;
    int status;
    int ifill[9];
} *sched_buf;
```

パラメータ

fbs_id

Frequency-Based Schedulerの識別子を表す一意の正の整数値。この値は**fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

sched_buf スケジュールしたいプロセスとスケジューリング・パラメータを含むpgm2_dsまたはpgm2_big_ds構造体。各コンポーネントで指定される情報の種類を以下に示します。*status*コンポーネントは本コールでは無視されることに注意して下さい：

name_ptr スケジュールされるプログラムを特定するパス名称を含む変数へのポインター。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。

cpu 指定したプログラムの実行をスケジュール可能なプロセッサーを示す整数値。許容値および対応する結果は次のとおり：

0 *name_ptr*で指定されたプログラムは呼び出しが行われるプロセッサーにスケジュールすることが可能です

-1 *name_ptr*で指定されたプログラムはどのプロセッサーでもスケジュールすることが可能です

bit mask (*cpu* & (1 << *i*))が設定された場合(*i*は0から15の範囲の整数でCPUを表します)、*name_ptr*で指定されたプログラムはCPU *i* にスケジュールすることが可能です。

cpuset 指定したプログラムの実行をスケジュール可能なプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

FBS_CPUSET_CURCPU
呼び出しが行われたプロセッサーでのみプロセスを実行することが可能です

FBS_CPUSET_ALLCPUS
全てのプロセッサーでプロセスを実行することが可能です

FBS_CPUSET_ONECPU(*cpu*)
CPU IDが*cpu*のプロセッサーでのみプロセスを実行することが可能です

*cpuset*が空の**cpuset(3)**を指し示している場合、その動作はFBS_CPUSET_CURCPUで説明しているとおりです。

fpid 戻される整数値はスケジュールされたプロセスに関する一意のFrequency-Based SchedulerプロセスIDです。

cid 指定したプログラムがスケジュールされるPOSIXスケジューリング・ポリシーを示す整数値。

スケジューリング・ポリシーはファイル**<sched.h>**に定義されています：SCHED_FIFO, SCHED_RR, SCHED_OTHER

prior 指定したプロセスのスケジューリング優先度を示す整数値。許容可能な優先度の値の範囲は指定するスケジューリング・ポリシーに影響されます。

param FBSでスケジュールされるプロセスに渡される整数値。この値は**sched_pgmqry**または**sched_pgmqry_big**(本ルーチンの説明は6-51ページを参照)の呼び出しを通してFBSにスケジュールされたプロセスにより読み出すことが可能です。

period 指定したプログラムが各メジャー・フレームで起動される頻度を示す整数値。

頻度1は指定したプログラムがマイナー・サイクル毎に起動されることを示し、頻度2はマイナー・サイクル2回毎に1回起動されることを示します。

この値は1から**fbsconfigure** (6-10ページを参照)の呼び出しで定義した時に指定したスケジューラーのフレームを構成するマイナー・サイクルの数の範囲とすることが可能です。

cycle 指定したプログラムが各フレームで起動するようにスケジュールされている最初のマイナー・サイクルを示す整数値。この値は0からフレームあたりのマイナー・サイクルの総数-1の範囲とすることが可能です。フレームあたりのマイナー・サイクルの総数は、**fbsconfigure** (6-10ページを参照)の呼び出しで指定されています。

halt 指定したプログラムがフレーム・オーバーランを引き起こした場合にスケジューラーを停止すべきかどうかを示す整数値。ゼロ以外の値はスケジューラーが停止されることを示します。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **sched_pgmadd(3)** または **sched_pgmadd_big(3)** の man ページを参照して下さい。

sched_pgmadd_args, sched_pgmadd_args_big - Frequency-Based Schedulerに引数を付けてプロセスをスケジュール

これらのルーチンは引数と共に新しいプロセスを生成して Frequency-Based Scheduler にそれをスケジュールします。新しいプロセスに引数とソフト・オーバーランおよび/またはデッドライン属性を提供するには、6-58 ページの **sched_pgmadd_attr** および **sched_pgmadd_attr_big** を参照して下さい。

これらのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int sched_pgmadd_args (fbs_id, sched_buf, argv[])
int fbs_id;
struct pgm2_ds {
    char *name_ptr;
    int cpu;
    int fpid;
    int cid;
    int prior;
    int param;
    int period;
    int cycle;
    int halt;
    int status;
} *sched_buf;
char *const argv[];

int sched_pgmadd_args_big (fbs_id, sched_buf, argv[])
int fbs_id;
struct pgm2_big_ds {
    char *name_ptr;
    cpuset_t cpuset;
    void *vfill[9];
    int fpid;
    int cid;
    int prior;
    int param;
    int period;
    int cycle;
    int halt;
    int status;
    int ifill[9];
} *sched_buf;
char *const argv[];
```

パラメータ

fbs_id

Frequency-Based Schedulerの識別子を表す一意の正の整数値。この値は**fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

sched_buf

スケジュールしたいプロセスとスケジューリング・パラメータを含むpgm2_dsまたはpgm2_big_ds構造体。各コンポーネントで指定される情報の種類を以下に示します。*status*コンポーネントは本コールでは無視されることに注意して下さい：

name_ptr

スケジュールされるプログラムを特定するパス名称を含む変数へのポインター。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。

cpu

指定したプログラムの実行をスケジュール可能なプロセッサーを示す整数値。許容値および対応する結果は次のとおり：

0
: *name_ptr*で指定されたプログラムは呼び出しが行われるプロセッサーにスケジュールすることが可能です

-1
: *name_ptr*で指定されたプログラムはどのプロセッサーでもスケジュールすることが可能です

bit mask
: (*cpu* & (1 << *i*))が設定された場合(*i*は0から15の範囲の整数でCPUを表します)、*name_ptr*で指定されたプログラムはCPU *i*にスケジュールすることが可能です。

cpuset

指定したプログラムの実行をスケジュール可能なプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

FBS_CPUSET_CURCPU
: 呼び出しが行われたプロセッサーでのみプロセスを実行することが可能です

FBS_CPUSET_ALLCPUS
: 全てのプロセッサーでプロセスを実行することが可能です

FBS_CPUSET_ONECPU(*cpu*)
: CPU IDが*cpu*のプロセッサーでのみプロセスを実行することが可能です

*cpuset*が空の**cpuset(3)**を指し示している場合、その動作はFBS_CPUSET_CURCPUで説明しているとおりです。

fpid 戻される整数値はスケジュールされたプロセスに関する一意のFrequency-Based SchedulerプロセスIDです。

cid 指定したプログラムがスケジュールされるPOSIXスケジューリング・ポリシーを示す整数値。スケジューリング・ポリシーはファイル**<sched.h>**に定義されています：SCHED_FIFO, SCHED_RR, SCHED_OTHER

prior 指定したプロセスのスケジューリング優先度を示す整数値。許容可能な優先度の値の範囲は指定するスケジューリング・ポリシーに影響されます。

param FBSでスケジュールされるプロセスに渡される整数値。この値は**sched_pgmqry**または**sched_pgmqry_big** (本ルーチンの説明は6-51ページを参照)の呼び出しを通してFBSにスケジュールされたプロセスにより読み出すことが可能です。

period 指定したプログラムが各メジャー・フレームで起動される頻度を示す整数値。

頻度1は指定したプログラムがマイナー・サイクル毎に起動されることを示し、頻度2はマイナー・サイクル2回毎に1回起動されることを示します。

この値は1から**fbsconfigure** (6-10ページを参照)の呼び出しで定義した時に指定したスケジューラーのフレームを構成するマイナー・サイクルの数の範囲とすることが可能です。

cycle 指定したプログラムが各フレームで起動するようにスケジュールされている最初のマイナー・サイクルを示す整数値。この値は0からフレームあたりのマイナー・サイクルの総数-1の範囲とすることが可能です。フレームあたりのマイナー・サイクルの総数は、**fbsconfigure** (6-10ページを参照)の呼び出しで指定されています。

halt 指定したプログラムがフレーム・オーバーランを引き起こした場合にスケジューラーを停止すべきかどうかを示す整数値。ゼロ以外の値はスケジューラーが停止されることを示します。

argv[] スケジュールされるプログラムで利用可能な引数のリストを表すNULLで終わる文字列へのポインターの配列。慣例により、最初の引数(argv[0])はスケジュールされるプログラムに関連するファイル名称を指し示すべきです。ポインターの配列はNULLポインターで終了する必要があります。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。

発生する可能性のあるエラーの種類を表示するには **sched_pgmadd_args(3)** または **sched_pgmadd_args_big(3)** の man ページを参照して下さい。

sched_pgmadd_attr, sched_pgmadd_attr_big - Frequency-Based Schedulerに引数と属性を付けてプロセスをスケジュール

これらのルーチンは引数とソフト・オーバーラン属性と共に新しいプロセスを生成して Frequency-Based Scheduler にそれをスケジュールします。新しいプロセスに対する指定したソフト・オーバーラン属性は、新しいプロセスが実行を開始する前に確実に初期化されます。

これらのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int sched_pgmadd_attr (fbs_id, sched_attr)
int fbs_id;
struct pgmadd_attr_ds *sched_attr;

struct pgmadd_attr_v1 {
    struct pgm2_ds *pgm2;
    char *const *argv;
    int so_soft_limit;
    int dl_halt;
    deadline_kind dl_kind;
    deadline_origin dl_origin;
    struct timespec dl_deadline;
};
struct pgmadd_attr_ds {
    int version;
    union {
        struct pgmadd_attr_v1 v1;
    } attr;
};

int sched_pgmadd_attr_big (fbs_id, sched_attr)
int fbs_id;
struct pgmadd_attr_big_ds *sched_attr;

struct pgmadd_attr_v1 {
    struct pgm2_ds *pgm2;
    char *const *argv;
    void *vfill[9];
    int so_soft_limit;
    int dl_halt;
    int ifill[9];
    deadline_kind dl_kind;
    deadline_origin dl_origin;
    struct timespec dl_deadline;
};
struct pgmadd_attr_big_ds {
    int version;
    union {
        struct pgmadd_attr_big_v1 v1;
```

```
    } attr;
};
```

パラメータ

fbs_id

Frequency-Based Schedulerの識別子を表す一意の正の整数値。この値は**`fbsid(3)`** (6-18ページ)または**`fbsconfigure(3)`** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

sched_attr

プロセスに対するスケジューリング・パラメータと属性を含む`pgmadd_attr_ds`または`pgmadd_attr_big_ds`構造体へのポインター。本構造体の各領域で指定される情報の種類を以下で説明します。

version

使用されている`attr`構造体のバージョン。現在バージョン1のみがサポートされているので、本領域は構造体`pgmadd_attr_v1`である*v1*が使用されていることを示すPGMADD_ATTR_VERSION1を設定する必要があります。

pgm2

プロセスをスケジュールするために使用されるスケジューリング・パラメータを含む`pgm2_ds`または`pgm2_big_ds`構造体へのポインター。`pgm2_ds`構造体の領域に関する詳細については6-54ページの**`sched_pgmadd_args`**項にある*sched_buf*の各パラメータを参照して下さい。

argv

引数が使用されない場合はNULLの値が設定されているポインター、さもなければスケジュールされるプログラムで利用可能な引数のリストを表すNULLで終わる文字列のポインター配列を指し示す必要のあるポインター。慣例により、最初の引数(argv[0])はスケジュールされるプログラムに関連するファイル名称を指し示すべきです。ポインターの配列はNULLポインターで終了する必要があります。

so_soft_limit

使用しない場合はゼロを設定、もしくはソフト・オーバーラン・プロセス上限値を含めることが可能です。ソフト・オーバーランの詳細については6-48ページの**`sched_pgm_set_soft_overrun_limit`**を参照して下さい。

dl_halt

プロセスでデッドライン違反が検出された場合にスケジューラーを停止させることを指定するにはDL_HALT、もしくはスケジューラーを停止させるべきではない場合にはDL_NOHALTを設定する必要があります。

dl_kind

スケジュールされるプロセスにデッドラインを適用しないことを示すDEADLINE_CLEAR、もしくはCLOCK_MONOTOICデッドライン時間を適用することを示すDEADLINE_WALL_TIMEを設定することが可能です。DEADLINE_WALL_TIMEを設定する場合、*dl_halt*, *dl_origin*, *dl_deadline*も設定される必要があります。

dl_origin

プロセスの実行がスケジュールされているサイクルの開始からデッドライン時間を測定するには

DL_CYCLE_REALTIME、もしくはプロセスが**fbswait(3)**を終了して実行を開始してからデッドライン時間を測定するにはDL_TASK_RELATIVEを設定することが可能です。

dl_deadline *dl_kind*にDEADLINE_WALL_TIMEが設定された場合、このtimespec構造体は**fbswait(3)**に戻る前にプロセスが実行するのに期待されるCLOCK_MONOTOIC時間を示します。詳細については6-45ページの**sched_pgm_set_deadlin**を参照して下さい。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために errno が設定されます)。発生する可能性のあるエラーの種類を表示するには **sched_pgmadd_attr(3)** または **sched_pgmadd_attr_big(3)** の man ページを参照して下さい。

sched_pgmqry, sched_pgmqry_big - プロセスを問合せ

これらのルーチンは Frequency-Based Scheduler にスケジュールされている特定のプロセスに関する情報を取得します。以下のいずれかの方法を使ってプロセスを特定することが可能です：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based SchedulerプロセスIDを指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based SchedulerプロセスIDを指定

NOTE

同じ CPU 上で複数回スケジュールされているプロセスを識別するために使用することが可能な唯一の方法は、その Frequency-Based Scheduler プロセス ID を指定することです。

返される情報は次を含みます：

- プロセスのパス名称
- プロセスを実行することが可能なCPU
- Frequency-Based SchedulerプロセスID
- スケジュールされているプロセスのスケジューリング・ポリシー
- スケジューリング優先度
- 周期(プロセスが各メジャー・フレームで起動される頻度を示すマイナー・サイクルの数)
- 開始ベース・サイクル(プロセスが各メジャー・フレームで起動される予定の最初のマイナー・サイクル)
- 「halt on overrun」フラグの値

これらのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int sched_pgmqry （fbs_id, qry_buf）
int fbs_id;
struct pgm2_ds {
     char *name_ptr;
     int cpu;
     int fpid;
     int cid;
     int prior;
     int param;
     int period;
     int cycle;
     int halt;
     int status;
} *qry_buf;
```

```
int sched_pgmqry_big （fbs_id, cpuset, fps_buf, buf_cnt）
int fbs_id, buf_cnt;
struct pgm2_big_ds {
    char *name_ptr;
    cpuset_t *cpuset;
    void *vfill[9];
    int fpid;
    int cid;
    int prior;
    int param;
    int period;
    int cycle;
    int halt;
    int status;
    int ifill[9];
} *qry_buf;
```

パラメータ

fbs_id Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

qry_buf プロセスに関するスケジューリング情報を含むpgm2_dsまたはpgm2_big_ds構造体へのポインター。**sched_pgmqry**は指定したプロセスのスケジューリング情報をこの構造体に戻します。*qry_buf*が指し示す構造体の各コンポーネントに含まれている情報を以下に示します：

name_ptr 情報が返されるプロセスを特定するパス名称へのポインター。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。ポインターがNULL文字列を指し示す場合、*fpid*コンポーネントにFrequency-Based SchedulerプロセスIDを提供する必要があります。

cpu 情報を取得するプログラムを特定するために*name_ptr*の値と共に使用されるプロセッサーを示す整数値。許容値および対応する結果は次のとおり：

0 呼び出しが行われたプロセッサー上で現在実行中の*name_ptr*が指し示す名称と名前が一致する最初のプロセスが要求されます。

-1 いずれかのプロセッサー上で現在実行中の*name_ptr*が指し示す名称と名前が一致する最初のプロセスが要求されます。

bit mask (*cpu* & (1 << *i*))が設定され(*i*は0から15の範囲の整数でCPUを表します)、それが

唯一のビット・セットである場合、CPU *i* で実行中の*name_ptr*が指し示す名称と名前が一致する最初のプロセスが要求されます

(*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name_ptr*が指し示す名称と名前が一致する最初のプロセスが要求されます

cpuset
: 問合せするプロセスを特定するため、*name_ptr*と関連して使用されるプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

`FBS_CPUSET_CURCPU`
: 呼び出しが行われたプロセッサーで現在実行中の*name_ptr*で指定された最初のプロセス

`FBS_CPUSET_ALLCPUS`
: 全てのプロセッサーで実行中の*name_ptr*で指定された最初のプロセス

`FBS_CPUSET_ONECPU`(*cpu*)
: CPU IDが*cpu*のプロセッサーで実行中の*name_ptr*で指定された最初のプロセス

*cpuset*が空の**cpuset(3)**を指し示している場合、その動作は`FBS_CPUSET_CURCPU`で説明しているとおりです。

fpid	情報が返されるプロセスに関するFrequency-Based Scheduler識別子を提供する整数値。この値は **sched_pgmadd** (6-51ページを参照)の呼び出しを行ったときに取得します。*name_ptr*と*cpu*を指定するだけで問い合わせてプログラムを特定したい場合、この値は-1とする必要があります。
cid	指定したプロセスのスケジューリング・ポリシーを示す整数値。
prior	プロセスのスケジューリング優先度を示す整数値。
param	**sched_pgmadd**または**sched_pgmresched**の呼び出しを通してプロセスに渡される値を示す整数値。
period	指定したプログラムが各メジャー・フレームで起動される頻度を示す整数値。
cycle	指定したプログラムが各フレームで起動するようにスケジュールされている最初のマイナー・サイクルを示す整数値。
halt	「halt on overrun」フラグの値を示す整数値。ゼロ以外の値はフラグが設定されることを示します。値が0はフラグが設定されないことを示します。
status	**<fbsched.h>**に定義されている指定したプロセスの現在の状況を示す整数値。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **sched_pgmqry(3)** または **sched_pgmqry_big(3)** の man ページを参照して下さい。

sched_pgmresched, sched_pgmresched_big - プロセスを再スケジュール

これらのルーチンは Frequency-Based Scheduler にスケジュールされているプロセスに関するスケジューリング・パラメータを変更します。例えば、プログラムのポリシーや優先度もしくは実行がスケジュールされている頻度を変更したくなる場合があります。しかしながら、スケジュールされた CPU を変更することは出来ません。

本ルーチンを(1)スケジューリング・ポリシーを SCHED_FIFO または SCHED_RR ポリシーへ変更、(2)SCHED_FIFO または SCHED_RR でスケジュールされたプロセスの優先度を変更、(3)プロセスごとの上限を超えて SCHED_OTHER でスケジュールされたプロセスの優先度を上げる、これらのために使用するには、呼び出し元プロセスの実効ユーザーID はターゲット・プロセス(スケジューリング・ポリシーと優先度が設定されているプロセス)の実効ユーザーID と一致している必要があることに注意することが重要です。

以下のいずれかの方法を使って再スケジュールするプロセスを特定することが可能です：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based Schedulerプロセス IDを指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based Schedulerプロセス IDを指定

NOTE

同じ CPU 上で複数回スケジュールされているプロセスを識別するために使用することが可能な唯一の方法は、その Frequency-Based Scheduler プロセス ID を指定することです。

これらのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int sched_pgmresched (fbs_id, rsch_buf)
int fbs_id;
struct pgm2_ds {
    char *name_ptr;
    int cpu;
    int fpid;
    int cid;
    int prior;
    int param;
    int period;
    int cycle;
    int halt;
    int status;
} *rsch_buf;

int sched_pgmresched_big (fbs_id, rsch_buf)
int fbs_id;
struct pgm2_big_ds {
    char *name_ptr;
    cpuset_t *cpuset;
```

```
    void *vfill[9];
    int fpid;
    int cid;
    int prior;
    int param;
    int period;
    int cycle;
    int halt;
    int status;
    int ifill[9];
} *rsch_buf;
```

パラメータ

fbs_id Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

rsch_buf プロセスを再スケジュールするためのスケジューリング・パラメータを含むpgm2_dsまたはpgm2_big_ds構造体。各コンポーネントで指定される情報の種類を以下に示します。*status*コンポーネントは本呼び出しでは無視されることに注意して下さい。

name_ptr 情報を再スケジュールするプロセスを特定するパス名称を含む変数へのポインター。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。ポインターがNULL文字列を指し示す場合、*fpid*コンポーネントにFrequency-Based Schedulerプロセスidを提供する必要があります。

cpu 再スケジュールするプロセスを特定するために*name_ptr*の値と共に使用されるプロセッサーを示す整数値。許容値および対応する結果は次のとおり：

0 呼び出しが行われたプロセッサー上で現在実行中の*name_ptr*が指し示す名称と名前が一致する最初のプロセスが再スケジュールされます。

-1 いずれかのプロセッサー上で現在実行中の*name_ptr*が指し示す名称と名前が一致する最初のプロセスが再スケジュールされます。

bit mask (*cpu* & (1 << *i*))が設定され(*i*は0から15の範囲の整数でCPUを表します)、それが唯一のビット・セットである場合、CPU *i*で実行中の*name_ptr*が指し示す名称と名前が一致する最初のプロセスが再スケジュールされます

(*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択した

CPUのいずれかで現在実行中の*name_ptr*が指し示す名称と名前が一致する最初のプロセスが再スケジュールされます

cpuset
再スケジュールするプロセスを特定するため、*name_ptr*と関連して使用されるプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

FBS_CPUSET_CURCPU
呼び出しが行われたプロセッサーで現在実行中の*name_ptr*で指定された最初のプロセス

FBS_CPUSET_ALLCPUS
全てのプロセッサーで実行中の*name_ptr*で指定された最初のプロセス

FBS_CPUSET_ONECPU(*cpu*)
CPU IDが*cpu*のプロセッサーで実行中の*name_ptr*で指定された最初のプロセス

*cpuset*が空の**cpuset(3)**を指し示している場合、その動作はFBS_CPUSET_CURCPUで説明しているとおりです。

fpid
再スケジュールされるプロセスに関するFrequency-Based SchedulerプロセスIDを提供する整数値。この値は**sched_pgmadd** (6-51ページを参照)の呼び出しを行ったときに取得します。*name_ptr*と*cpu*を指定するだけで問い合わせてプログラムを特定したい場合、この値は-1とする必要があります。

cid
指定したプロセスのスケジューリング・ポリシーを示す整数値：SCHED_FIFO, SCHED_RR, SCHED_OTHER

prior
プロセスのスケジューリング優先度を示す整数値。許容可能な優先度の値の範囲は指定するスケジューリング・ポリシーに影響されます。

param
Frequency-Based Schedulerにスケジュールされたプロセスに渡される値を示す整数値。

period
指定したプログラムが各メジャー・フレームで起動される頻度を示す整数値。

頻度1は指定したプログラムがマイナー・サイクル毎に起動されることを示し、頻度2はマイナー・サイクル2回毎に1回起動されることを示します。

この値は1から**fbsconfigure** (6-10ページを参照)の呼び出しで定義した時に指定したスケジューラーのフレームを構成するマイナー・サイクルの数の範囲とすることが可能です。

cycle 指定したプロセスが各フレームで起動するようにスケジュールされている最初のマイナー・サイクルを示す整数値。

この値は0からフレームあたりのマイナー・サイクルの総数-1の範囲とすることが可能です。フレームあたりのマイナー・サイクルの総数は、**fbsconfigure** (6-10ページを参照)の呼び出しで指定されています。

halt 指定したプロセスがフレーム・オーバーランを引き起こした場合にスケジューラーを停止すべきかどうかを示す整数値。ゼロ以外の値はスケジューラーが停止されることを示します。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **sched_pgmresched(3)** または **sched_pgmresched_big(3)** の man ページを参照して下さい。

Performance Monitorルーチン

Performance Monitor ルーチンは Performance Monitor の主要な機能へのアクセスを提供します。それらは以下のような基本的な操作を実行することを可能にします：

- プロセスまたはプロセッサーのPerformance Monitor値を消去
- プロセスまたはプロセッサーのPerformance Monitorを開始および停止
- プロセスまたはプロセッサーのPerformance Monitor値を取得

ルーチンの要約

Performance Monitor ルーチンが表 6-3 に要約されています。各ルーチンに関する完全な情報は「Performance Monitor ルーチンの使用」項で提供されます。

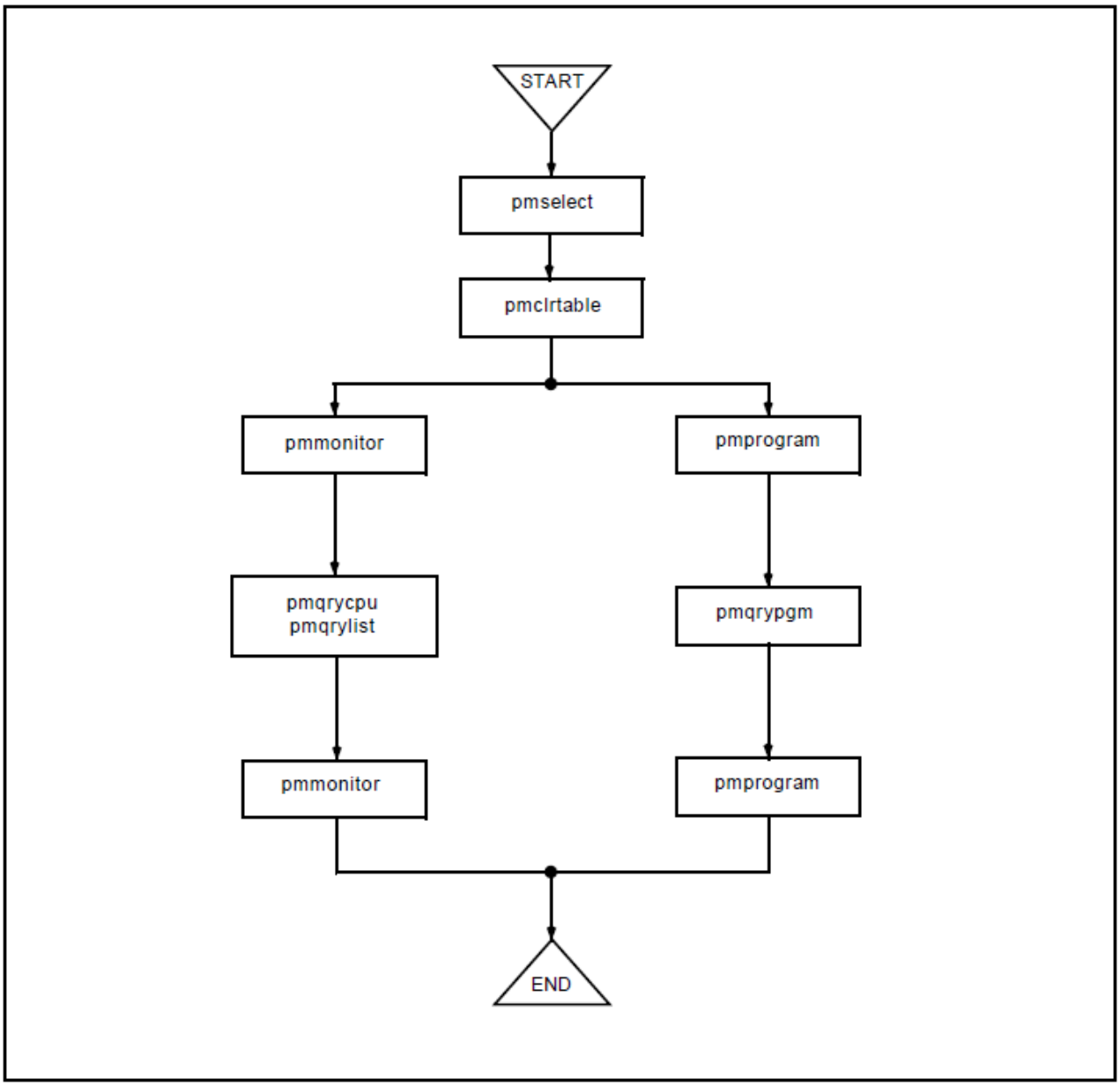
表6-3 Performance MonitorのCライブラリ・ルーチン

ルーチン	ページ	説明
pmclrpgm pmclrpgm_big	6-71	プロセスの値を消去
pmclrtable pmclrtable_big	6-73	プロセッサーの値を消去
pmmonitor pmmonitor_big	6-75	プロセッサーのPerformance Monitorを開始/停止
pmprogram pmprogram_big	6-77	プロセスのPerformance Monitorを開始/停止
pmqrycpu pmqrycpu_big	6-79	選択したプロセッサーの値を問合せ
pmqrylist	6-83	プロセスのリストの値を問合せ
pmqrypgm pmqrypgm_big	6-85	選択したプロセスの値を問合せ
pmqrytimer	6-88	Performance Monitorのモードを問合せ
pmselect	6-89	Performance Monitorのモードを選択

Cライブラリ・コールの順序

アプリケーション・プログラムから Performance Monitor ルーチンを呼び出す可能性のあるおおよその順番を図 6-2 に示します。

図6-2 Cライブラリ・コールの順序：Performance Monitor



Performance Monitorの使用

これ以降は、**libccur_fbsched** ライブラリ内に含まれる Performance Monitor ルーチンをアルファベット順で示します。

pmclrpgm, pmclrpgm_big - プロセスの値を消去

これらのルーチンは Frequency-Based Scheduler にスケジュールされた特定のプロセスの Performance Monitor の値を消去します。以下の方法のいずれかを使ってプロセスを特定することが可能です：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based SchedulerプロセスIDを指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based SchedulerプロセスIDを指定

NOTE

本ルーチンはプロセスのソフト・オーバーラン総数を消去します。

これらのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int pmclrpgm (fbs_id, name, cpu, fpid)
int fbs_id, cpu, fpid;
char *name;

int pmclrpgm_big (fbs_id, name, cpuset, fpid)
int fbs_id, fpid;
char *name;
cpuset_t *cpuset;
```

パラメータ

fbs_id Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

name 値を消去するプロセスを特定するパス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。本変数がNULL文字列である場合、*fpid*パラメータにFrequency-Based SchedulerプロセスIDを提供する必要があります。

cpu 値を消去するプロセスを特定するために*name*パラメータの値と関連して使用されるプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
-1	全てのプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます

bit mask
: (*cpu* & (1 << *i*))が設定され(*i*は0から15の範囲の整数でCPUを表します)、それが唯一のビット・セットである場合、CPU *i*で実行中の*name*で指定された最初のプロセスが指定されます

  (*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name*で指定された最初のプロセスが指定されます

cpuset
: 値を消去するプロセスを特定するため、*name*と関連して使用されるプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

  `FBS_CPUSET_CURCPU`
  : 呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセス

  `FBS_CPUSET_ALLCPUS`
  : 全てのプロセッサーで実行中の*name*で指定された最初のプロセス

  `FBS_CPUSET_ONECPU`(*cpu*)
  : CPU IDが*cpu*のプロセッサーで実行中の*name*で指定された最初のプロセス

  *cpuset*が空の**cpuset(3)**を指し示している場合、その動作は`FBS_CPUSET_CURCPU`で説明しているとおりです。

fpid
: 値を消去するプロセスに関する一意のFrequency-Based Schedulerプロセスを提供する整数値。この値は**sched_pgmadd** (6-51ページを参照)の呼び出しを行ったときに取得します。*name*と*cpu*を指定するだけでプロセスを特定するにはこの値は-1にする必要があります。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **pmclrpgm(3)** および **pmclrpgm_big(3)** の man ページを参照して下さい。

pmclrtable, pmclrtable_big - プロセッサーの値を消去

これらのルーチンは、選択したスケジューラーにおいて指定した 1 つ以上のプロセッサー上の FBS でスケジュールされたプロセスの Performance Monitor の値を消去します。

NOTE

本ルーチンは全ての関連するプロセスのソフト・オーバーラン総数を消去します。

これらのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int pmclrtable （fbs_id, cpu）
int fbs_id, cpu;

int pmclrtable_big （fbs_id, cpuset）
int fbs_id;
cpuset_t *cpuset;
```

パラメータ

fbs_id — Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

cpu — Performance Monitorの値を消去するプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで実行中のFBSでスケジュールされたプロセスに関するPerformance Monitorの値が消去されます
-1	スケジューラ上の全てのプロセスに関するPerformance Monitorの値が消去されます
bit mask	(*cpu* & (1 << *i*))が設定(*i*は0から15の範囲の整数でCPUを表します)されている場合、CPU *i*で実行中のプロセスのPerformance Monitorの値が消去されます

cpuset — Performance Monitorの値を消去するプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc (3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

`FBS_CPUSET_CURCPU`
呼び出しが行われたプロセッサー

`FBS_CPUSET_ALLCPUS`
全てのプロセッサー

`FBS_CPUSET_ONECPU(`*cpu*`)`
CPU IDが*cpu*のプロセッサー

*cpuset*が空の**`cpuset(3)`**を指し示している場合、その動作は`FBS_CPUSET_CURCPU`で説明しているとおりです。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには**`pmclrtable(3)`**と**`pmclrtable_big(3)`**の man ページを参照して下さい。

pmmonitor, pmmonitor_big - プロセッサーのPerformance Monitorを開始/停止

これらのルーチンは、選択したスケジューラーにおいて 1 つ以上のプロセッサー上に FBS でスケジュールされたプロセスの Performance Monitor を開始または停止します。

これらの 2 つのルーチンは CPU マスクの処理を除けば全く同じです。これらの違いの詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int pmmonitor （fbs_id, pmflag, cpu）
int fbs_id, pmflag, cpu;

int pmmonitor_big （fbs_id, pmflag, cpuset）
int fbs_id, pmflag;
cpuset_t *cpuset;
```

パラメータ

fbs_id
: Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

  識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

pmflag
: Performance Monitorを開始するのか停止するのかを示す整数値。ゼロ以外の値はPerformance Monitorを開始することを示します。値ゼロはPerformance Monitorを停止することを示します。

cpu
: Performance Monitorを開始または停止するプロセッサを示す整数値。許容値および対応する結果は次のとおり：

  0
  : 呼び出しが行われたプロセッサーで実行中のFBSでスケジュールされたプロセスに対してPerformance Monitorが開始または停止されます

  -1
  : スケジューラ上の全てのプロセスに関するPerformance Monitorが開始または停止されます

  bit mask
  : (*cpu* & (1 << *i*))が設定(*i*は0から15の範囲の整数でCPUを表します)されている場合、CPU *i*で実行中のプロセスに対してPerformance Monitorが開始または停止されます

cpuset
: Performance Monitorが開始または停止されるプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは **cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

`FBS_CPUSET_CURCPU`
呼び出しが行われたプロセッサー

`FBS_CPUSET_ALLCPUS`
全てのプロセッサー

`FBS_CPUSET_ONECPU(`*cpu*`)`
CPU IDが*cpu*のプロセッサー

*cpuset*が空の**`cpuset(3)`**を指し示している場合、その動作は`FBS_CPUSET_CURCPU`で説明しているとおりです。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **`pmmonitor(3)`** と **`pmmonitor_big(3)`** の man ページを参照して下さい。

pmprogram, pmprogram_big - プロセスのPerformance Monitorを開始/開始

これらのルーチンは Frequency-Based Scheduler にスケジュールされた特定のプロセスの Performance Monitor を開始または停止します。以下の方法のいずれかを使ってプロセスを特定して下さい：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based SchedulerプロセスIDを指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based SchedulerプロセスIDを指定

これらの 2 つのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int pmprogram （fbs_id, name, cpu, fpid, pmflag）
int fbs_id, cpu, fpid, pmflag;
char *name;

int pmprogram_big （fbs_id, name, cpuset, fpid, pmflag）
int fbs_id, fpid, pmflag;
char *name;
cpuset_t *cpuset;
```

パラメータ

fbs_id — Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

name — Performance Monitorを開始または停止するプロセスを特定するパス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。本変数がNULL文字列である場合、*fpid*パラメータにFrequency-Based SchedulerプロセスIDを提供する必要があります。

cpu — Performance Monitorを開始または停止するプロセスを特定するために*name*パラメータの値と関連して使用されるプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
-1	全てのプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
bit mask	(*cpu* & (1 << *i*))が設定され(*i*は0から15の範囲の整数でCPUを表します)、それが唯一のビット・セットである場合、CPU *i*で実行中の*name*で指定された最初のプロセスが指定されます

(*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name*で指定された最初のプロセスが指定されます

cpuset
: Performance Monitorを開始または停止するプロセスを特定するため、*name*と関連して使用されるプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

`FBS_CPUSET_CURCPU`
: 呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセス

`FBS_CPUSET_ALLCPUS`
: 全てのプロセッサーで実行中の*name*で指定された最初のプロセス

`FBS_CPUSET_ONECPU(`*cpu*`)`
: CPU IDが*cpu*のプロセッサーで実行中の*name*で指定された最初のプロセス

*cpuset*が空の**cpuset(3)**を指し示している場合、その動作は`FBS_CPUSET_CURCPU`で説明しているとおりです。

fpid
: Performance Monitorを開始または停止するプロセスに関する一意のFrequency-Based SchedulerプロセスIDを提供する整数値。

この値は**sched_pgmadd** (6-51ページを参照)の呼び出しを行って取得して下さい。*name*と*cpu*を指定するだけでプロセスを特定するにはこの値は-1にする必要があります。

pmflag
: Performance Monitorを開始するのか停止するのかを示す整数値。ゼロ以外の値はPerformance Monitorを開始することを示します。値ゼロはPerformance Monitorを停止することを示します。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **pmprogram(3)** と **pmprogram_big(3)** の man ページを参照して下さい。

pmqrycpu, pmqrycpu_big - 選択したプロセッサーの値を問合せ

これらのルーチンは、選択したスケジューラーにおいて 1 つ以上のプロセッサー上に FBS でスケジュールされたプロセスの Performance Monitor の値を取得します。

これらの 2 つのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int pmqrycpu （fbs_id, cpu, pm_buf, buf_cnt）
int fbs_id, cpu, buf_cnt;
struct pmqry_ds {
    int                     fpid;
    struct timespec         lastcyc_tm;
    int                     tot_cycles;
    struct timespec         tot_cycles_tm;
    int                     overruns;
    struct timespec         mincyc_tm;
    int                     mincyc_cycle;
    int                     mincyc_frame;
    struct timespec         maxcyc_tm;
    int                     maxcyc_cycle;
    int                     maxcyc_frame;
    struct timespec         minframe_tm;
    int                     minframe;
    struct timespec         maxframe_tm;
    int                     maxframe;
} *pm_buf;

int pmqrycpu_big （fbs_id, cpuset, pm_buf, buf_cnt）
int fbs_id, buf_cnt;
cpuset_t *cpuset;
struct pmqry_ds *pm_buf;
```

パラメータ

fbs_id

Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

cpu

Performance Monitorの値を取得するプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで実行中のFBSでスケジュールされたプロセスに関するPerformance Monitorの値が返されます
-1	スケジューラ上の全てのプロセスに関するPerformance Monitorの値が返されます

bit mask　(*cpu* & (1 << *i*))が設定(*i*は0から15の範囲の整数でCPUを表します)されている場合、CPU *i*で実行中のプロセスのPerformance Monitorの値が返されます

cpuset　Performance Monitorの値を取得するプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効なcpusetを指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

FBS_CPUSET_CURCPU
　呼び出しが行われたプロセッサー

FBS_CPUSET_ALLCPUS
　全てのプロセッサー

FBS_CPUSET_ONECPU(*cpu*)
　CPU IDが*cpu*のプロセッサー

*cpuset*が空の**cpuset(3)**を指し示している場合、その動作はFBS_CPUSET_CURCPUで説明しているとおりです。

pm_buf　**pmqrycpu**が*cpu*パラメータで指定したプロセッサ上のFBSでスケジュールされた各プロセスに関するPerformance Monitorの値を返すpmqry_ds構造体の配列へのポインター。

これらの値が返されるプロセスの数は*buf_cnt*パラメータの値により制約されます。単一プロセスに関する構造体の各コンポーネントに返される情報の種類を以下に示します：

fpid　プロセスのFrequency-Based Schedulerプロセス ID

lastcyc_tm　プロセスが最後にスケジューラーに起動されてから**fbswait**が呼ばれるまでに実行に費やした時間

tot_cycles　プロセスがスケジューラーに起動された回数(合計反復数、またはサイクル)

tot_cycles_tm　プロセスが全てのサイクルで実行に費やした時間

overruns　プロセスに起因するハード・フレーム・オーバーランの数

mincyc_tm　プロセスが1サイクルで実行に費やした時間の最小(最小サイクル時間)

mincyc_cycle
　最小サイクル時間が発生したマイナー・サイクルの番号(最小サイクル・サイクル)

mincyc_frame
　最小サイクル時間が発生したメジャー・フレームの番号(最小サイクル・フレーム)

maxcyc_tm	プロセスが1サイクルで実行に費やした時間の最大(最大サイクル時間)
maxcyc_cycle	最大サイクル時間が発生したマイナー・サイクルの番号(最大サイクル・サイクル)
maxcyc_frame	最大サイクル時間が発生したメジャー・フレームの番号(最大サイクル・フレーム)
minframe_tm	プロセスが1メジャー・フレームの間で実行に費やした時間の最小(最小フレーム時間)
minframe	最小フレーム時間が発生したメジャー・フレームの番号(最小フレーム・フレーム)
maxframe_tm	プロセスが1メジャー・フレームの間で実行に費やした時間の最大(最大フレーム時間)
maxframe	最大フレーム時間が発生したメジャー・フレームの番号(最大フレーム・フレーム)

buf_cnt *pm_buf*が示す配列内の構造体の数を示す整数値。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **`pmqrycpu(3)`** と **`pmqrycpu_big(3)`** の man ページを参照して下さい。

pmqrylist - プロセスのリストの値を問合せ

本ルーチンは、Frequency-Based Scheduler 上にスケジュールされたプロセスのリストの Performance Monitor の値を取得します。

概要

```
#include <fbsched.h>

int pmqrylist (fbs_id, pm_buf, buf_cnt)
int fbs_id, buf_cnt;
struct pmqry_ds {
    int                 fpid;
    struct timespec     lastcyc_tm;
    int                 tot_cycles;
    struct timespec     tot_cycles_tm;
    int                 overruns;
    struct timespec     mincyc_tm;
    int                 mincyc_cycle;
    int                 mincyc_frame;
    struct timespec     maxcyc_tm;
    int                 maxcyc_cycle;
    int                 maxcyc_frame;
    struct timespec     minframe_tm;
    int                 minframe;
    struct timespec     maxframe_tm;
    int                 maxframe;
} *pm_buf;
```

パラメータ

fbs_id

Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

pm_buf

pmqrylistがFBSでスケジュールされたプロセスのリストに関するPerformance Monitorの値を返すpmqry_ds構造体の配列へのポインター。

値が返されるプロセスのリストは、配列内の各構造体の*fpid*コンポーネントにFrequency-Based Scheduler識別子を設定することで生成されます。単一プロセスに関する構造体の各コンポーネントに含まれる情報の種類を以下に示します：

fpid　プロセスのFrequency-Based SchedulerプロセスID

lastcyc_tm　プロセスが最後にスケジューラーに起動されてから**fbswait**が呼ばれるまでに実行に費やした時間

tot_cycles　プロセスがスケジューラーに起動された回数(合計反復数、またはサイクル数)

tot_cycles_tm	プロセスが全てのサイクルで実行に費やした時間
overruns	プロセスに起因するハード・フレーム・オーバーランの数
mincyc_tm	プロセスが1サイクルで実行に費やした時間の最小(最小サイクル時間)
mincyc_cycle	最小サイクル時間が発生したマイナー・サイクルの番号(最小サイクル・サイクル)
mincyc_frame	最小サイクル時間が発生したメジャー・フレームの番号(最小サイクル・フレーム)
maxcyc_tm	プロセスが1サイクルで実行に費やした時間の最大(最大サイクル時間)
maxcyc_cycle	最大サイクル時間が発生したマイナー・サイクルの番号(最大サイクル・サイクル)
maxcyc_frame	最大サイクル時間が発生したメジャー・フレームの番号(最小サイクル・フレーム)
minframe_tm	プロセスが1メジャー・フレームの間で実行に費やした時間の最小(最小フレーム時間)
minframe	最小フレーム時間が発生したメジャー・フレームの番号(最小フレーム・フレーム)
maxframe_tm	プロセスが1メジャー・フレームの間で実行に費やした時間の最大(最大フレーム時間)
maxframe	最大フレーム時間が発生したメジャー・フレームの番号(最大フレーム・フレーム)

buf_cnt *pm_buf*が示す配列内の構造体の数を示す整数値。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **`pmqrylist(3)`** の man ページを参照して下さい。

pmqrypgm, pmqrypgm_big - 選択したプロセスの値を問合せ

これらのルーチンは、Frequency-Based Scheduler にスケジュールされた特定のプロセスの Performance Monitor の値を取得します。以下の方法のいずれかを使ってプロセスを特定して下さい：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based Schedulerプロセスを指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based Schedulerプロセスを指定

これらの 2 つのルーチンは CPU マスクの処理を除けば全く同じです。詳細については、6-2 ページの「Big-SMP FBS インターフェース」を参照して下さい。

概要

```
#include <fbsched.h>

int pmqrypgm (fbs_id, name, cpu, pm_buf)
int fbs_id, cpu;
char *name;
struct pmqry_ds {
    int                 fpid;
    struct timespec     lastcyc_tm;
    int                 tot_cycles;
    struct timespec     tot_cycles_tm;
    int                 overruns;
    struct timespec     mincyc_tm;
    int                 mincyc_cycle;
    int                 mincyc_frame;
    struct timespec     maxcyc_tm;
    int                 maxcyc_cycle;
    int                 maxcyc_frame;
    struct timespec     minframe_tm;
    int                 minframe;
    struct timespec     maxframe_tm;
    int                 maxframe;
} *pm_buf;

int pmqrypgm_big (fbs_id, name, cpuset, pm_buf)
int fbs_id;
char *name;
cpuset_t *cpuset;
struct pmqry_ds *pm_buf;
```

パラメータ

fbs_id
: Frequency-Based Schedulerの識別子を表す一意の正の整数。この値は **fbsid(3)** (6-18ページ)または**fbsconfigure(3)** (6-10ページ)を使って取得することが可能です。

  識別子を認識せずに呼び出しプロセスがスケジュールされているFrequency-Based Schedulerを参照するにはMY_FBSを入力して下さい。

name
: Performance Monitorの値を返すプロセスを特定するパス名称を含む変数へのポインター。

最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。本変数がNULL文字列である場合、*pm_buf*が示す構造体の`fpid`コンポーネントにFrequency-Based Schedulerプロセスを提供する必要があります。

cpu

Performance Monitorの値を返すプロセスを特定するために*name*パラメータの値と関連して使用されるプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
-1	全てのプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
bit mask	(*cpu* & (1 << *i*))が設定され(*i*は0から15の範囲の整数でCPUを表します)、それが唯一のビット・セットである場合、CPU *i*で実行中の*name*で指定された最初のプロセスが指定されます
	(*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name*で指定された最初のプロセスが指定されます

cpuset

Performance Monitorの値を返すプロセスを特定するため、*name*と関連して使用されるプロセッサーのセットを直接的または間接的に示すポインター。そのポインターは**cpuset_alloc(3)**に生成されるか、他の**cpuset(3)**操作で何かしらのCPU IDのセットが満たされるかのどちらかの有効な`cpuset`を指し示す、または次の特殊なポインター値のいずれかを選ぶことが可能です：

`FBS_CPUSET_CURCPU`
: 呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセス

`FBS_CPUSET_ALLCPUS`
: 全てのプロセッサーで実行中の*name*で指定された最初のプロセス

`FBS_CPUSET_ONECPU(`*cpu*`)`
: CPU IDが*cpu*のプロセッサーで実行中の*name*で指定された最初のプロセス

*cpuset*が空の**cpuset(3)**を指し示している場合、その動作は`FBS_CPUSET_CURCPU`で説明しているとおりです。

pm_buf

pmqrypgmが*name*パラメータで指定されたFBSでスケジュールされた各プロセスに関するPerformance Monitorの値を返す`pmqry_ds`構造体へのポインター。

構造体の各コンポーネントに含まれる情報の種類を以下に示します：

fpid	Performance Monitorの値を返す一意のFrequency-Based SchedulerプロセスIDを提供する整数値。 この値は**sched_pgmadd** (6-51ページを参照)の呼び出しを行って取得して下さい。*name*と*cpu*を指定するだけでプロセスを特定するにはこの値は-1にする必要があります。
lastcyc_tm	プロセスが最後にスケジューラーに起動されてから**fbswait**が呼ばれるまでに実行に費やした時間
tot_cycles	最後にPerformance Monitorの値が消去されPerformance Monitorが有効になってからプロセスがスケジューラーに起動された回数(合計反復数、またはサイクル数)
tot_cycles_tm	プロセスが全てのサイクルで実行に費やした時間
overruns	プロセスに起因するハード・フレーム・オーバーランの数
mincyc_tm	プロセスが1サイクルで実行に費やした時間の最小(最小サイクル時間)
mincyc_cycle	最小サイクル時間が発生したマイナー・サイクルの番号(最小サイクル・サイクル)
mincyc_frame	最小サイクル時間が発生したメジャー・フレームの番号(最小サイクル・フレーム)
maxcyc_tm	プロセスが1サイクルで実行に費やした時間の最大(最大サイクル時間)
maxcyc_cycle	最大サイクル時間が発生したマイナー・サイクルの番号(最大サイクル・サイクル)
maxcyc_frame	最大サイクル時間が発生したメジャー・フレームの番号(最小サイクル・フレーム)
minframe_tm	プロセスが1メジャー・フレームの間で実行に費やした時間の最小(最小フレーム時間)
minframe	最小フレーム時間が発生したメジャー・フレームの番号(最小フレーム・フレーム)
maxframe_tm	プロセスが1メジャー・フレームの間で実行に費やした時間の最大(最大フレーム時間)
maxframe	最大フレーム時間が発生したメジャー・フレームの番号(最大フレーム・フレーム)

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **pmqrypgm(3)** と **pmqrypgm_big(3)** の man ページを参照して下さい。

pmqrytimer - Performance Monitorのモードを問合せ

本ルーチンは、Performance Monitor のタイミング値が割り込みの処理に費やす時間を含めるのか除外するのかを判定します。

概要

```
#include <fbsched.h>

int pmqrytimer ()
```

戻り値

戻り値 0 は割り込み時間が Performance Monitor のタイミング値から除外されることを示します。戻り値 1 は割り込み時間がタイミング値に含まれることを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **`pmqrytimer(3)`** の man ページを参照して下さい。

pmselect - Performance Monitorのモードを選択

本ルーチンは、Performance Monitor を実行するタイミング・モードを選択します。タイミング・モードは割り込みを処理するのに費やした時間を含めるもしくは除外することを設定することが可能です。

割り込み時間を含んでいる場合、プロセスのユーザーおよびシステム時間は、プロセスが現在処理を実行している時に割り込みを処理するのに費やした全ての時間を含めて生じた経過時間を合計します。割り込みを処理するのに費やした時間はプロセスのシステム時間に加算されます。

割り込み時間を除外している場合、プロセスのユーザーおよびシステム時間は、プロセスが現在処理を実行している時に割り込みを処理するのに費やした全ての時間を除外して生じた時間を合計します。

概要

```
#include <fbsched.h>

int pmselect (mode)
int mode;
```

パラメータ

mode　割り込みを処理するのに費やした時間をPerformance Monitorのタイミング値に含めるのか除外するのかを示す整数値。

ゼロ以外の値は割り込み時間が含まれることを示します。値ゼロは割り込み時間が除外されることを示します。

戻り値

戻り値 0 は呼び出しが成功したことを示します。戻り値-1 はエラーが発生したことを示します(エラーを示すために `errno` が設定されます)。発生する可能性のあるエラーの種類を表示するには **`pmselect(3)`** の man ページを参照して下さい。

7章

FORTRANライブラリ・インターフェース

概要 7-1
プログラムのコンパイルとリンク 7-1
Big-SMP FBSインターフェース 7-2
Frequency-Based Schedulerルーチン 7-3
ルーチンの要約 7-3
FORTRANライブラリ・コールの順序 7-5
Frequency-Based Schedulerルーチンの使用 7-6
fbsaccess – Frequency-Based Schedulerのアクセス権を変更 7-6
fbsattach – Frequency-Based Schedulerにタイミング・ソースを接続 7-8
fbsconfigure – Frequency-Based Schedulerを構成 7-9
fbscycle – マイナー・サイクル/メジャー・フレームの総数を返答 7-11
fbsdetach – Frequency-Based Schedulerからタイミング・ソースを分離 7-12
fbsgetrtc – リアルタイム・クロックの現在値を取得 7-13
fbsid – キーに対するFrequency-Based Scheduler識別子を返答 7-14
fbsinfo – Frequency-Based Schedulerの情報を返答 7-15
fbsintrpt – スケジューリングを開始/停止/再開 7-17
fbsquery – Frequency-Based Schedulerのプロセスを問合せ 7-18
fbsremove – Frequency-Based Schedulerを削除 7-21
fbsresume – Frequency-Based Schedulerのスケジューリングを再開 7-22
fbsrunrtc – リアルタイム・クロックを開始/停止 7-24
fbsschedself – Frequency-Based Schedulerにプロセス/スレッドをスケジュール . . . 7-25
fbssetrtc – リアルタイム・クロックを設定 7-27
fbswait – Frequency-Based Schedulerを待機 7-28
nametopid – 指定したプロセス名称のプロセスIDを返答 7-29
pgmquery – Frequency-Based Schedulerのプロセスを問合せ 7-30
pgmremove – Frequency-Based Schedulerからプロセスを削除 7-32
pgmreschedule – プロセスを再スケジュール 7-34
pgmschedule – Frequency-Based Schedulerにプロセスをスケジュール 7-37
pgmstat – FBSでスケジュールされたプロセスの状態を問合せ 7-40
pgmtrigger – FBSの待ち状態プロセスを起動 7-42
rtparm – 初期パラメータを返答 7-43
sched_pgm_deadline_query – プロセスに割り当て済みデッドラインを問合せ . . . 7-44
sched_pgm_deadline_test – デッドライン違反の有無をテスト 7-46
sched_pgm_set_deadline – プロセス・デッドライン時間を設定または消去 7-48
sched_pgm_set_soft_overrun_limit – ソフト・オーバーラン上限を設定 7-50
sched_pgm_soft_overrun_query – ソフト・オーバーラン処理を問合せ 7-51
schedfbsqry – Frequency-Based Schedulerのプロセスを問合せ 7-52
schedpgmadd – Frequency-Based Schedulerのプロセスをスケジュール 7-55
schedpgmadd_args – Frequency-Based Schedulerに引数を付けてプロセスをスケジュール 7-57
schedpgmqry – Frequency-Based Schedulerのプロセスを問合せ 7-59
schedpgmresched – プロセスを再スケジュール 7-62
Performance Monitorルーチン 7-65
ルーチンの要約 7-65
FORTRANライブラリ・コールの順序 7-65
Performance Monitorルーチンの使用 7-67
pmclrpgm, – プロセスの値を消去 7-67
pmclrtable – プロセッサーの値を消去 7-69

pmmonitor – プロセッサーのPerformance Monitorを開始/停止 7-70
pmprogram – プロセスのPerformance Monitorを開始/停止 7-71
pmqrycpu – 選択したプロセッサーの値を問合せ 7-73
pmqrylist – プロセスのリストの値を問合せ 7-75
pmqrypgm – 選択したプロセスの値を問合せ 7-77
pmquerytimer – Performance Monitorのモードを問合せ 7-80
pmselect – Performance Monitorのモードを選択 7-81

7章

FORTRANライブラリ・インターフェース

本章では Frequency-Based Scheduler と Performance Monitor への FORTRAN ライブラリ・インターフェースについて説明します。ライブラリ情報、スケジューラーおよび Performance Monitor の呼び出し順序、各々のライブラリ・ルーチンの詳細を提供します。

概要

Frequency-Based Scheduler と Performance Monitor に関連付けられた機能へのアクセスは FORTRAN で書かれたアプリケーションから呼び出すことが可能なルーチンのライブラリを通して提供されます。

各ルーチンについて以下の情報が本章内で提供されます：

- ルーチンの説明
- アプリケーション・プログラム内でルーチンを参照するために必要となるFORTRAN変数宣言や呼出し命令
- 各パラメータの詳細な説明

FBS でスケジューリング中にルーチンを呼び出す順序の図を含む個々のルーチンに関する情報は 7-3 ページから始まります。Performance Monitor に関する同じ情報が 7-65 ページから提供されます。

PowerMAX オペレーティング・システムとの互換性のため、**usermap(3f)** がアドレス空間にターゲット・プロセスのメモリをマッピングするために提供されます。詳細は man ページを参照して下さい。

プログラムのコンパイルとリンク

FORTRAN プログラムを静的にリンクする場合、以下のライブラリが必要となります：

/usr/lib/libccur_fbsched.a または**/usr/lib64/libccur_fbsched.a**
/usr/lib/libccur_rt.a または**/usr/lib64/libccur_rt.a**
/usr/lib/libF77rt.a または**/usr/lib64/libF77rt.a**

FORTRAN プログラムを動的にリンクする場合、以下のライブラリが必要となります：

/usr/lib/libccur_fbsched.so または**/usr/lib64/libccur_fbsched.so**
/usr/lib/libccur_rt.so または**/usr/lib64/libccur_rt.so**
/usr/lib/libF77rt.so または**/usr/lib64/libF77rt.so**

FORTRAN プログラムをコンパイルしてリンクするには、コマンド・ライン命令は次のとおりです：

cf77 [*options...*] *file*.**f -lF77rt -lccur_fbsched -lccur_rt**

Big-SMP FBSインターフェース

FORTRAN 言語はまだ Big-SMP FBS インターフェースの API は提供されていません。従いまして、Big-SMP API へのアクセスが必要となる FORTRAN アプリケーションは間接的(例えば、**rtcp(1)** コマンドを通して FBS サービスを起動、もしくはそのアプリケーションで要求される Big-SMP FBS サービスを起動する C ルーチンを書いてリンク)に行う必要があります。

Frequency-Based Schedulerルーチン

Frequency-Based Scheduler ルーチンはスケジューラーの主要な機能へのアクセスを提供します。それらは以下のような基本的な操作を実行することを可能にします：

- スケジューラーを構成
- スケジューラーにプログラムをスケジュール
- スケジューラーにタイミング・ソースを設定および接続
- スケジューラーのスケジュールを開始、停止、再開
- スケジュールされたプロセスに関する情報を取得
- スケジュールされたプロセスを再スケジュールおよび削除
- タイミング・ソースを分離
- スケジューラーの削除

ルーチンの要約

Frequency-Based Scheduler ルーチンが表 7-1 に要約されています。各ルーチンに関する完全な情報は「Frequency-Based Scheduler ルーチンの使用」項で提供されます。

表7-1 Frequency-Based SchedulerのFORTRANライブラリ・ルーチン

ルーチン	ページ	説明
fbsaccess	7-6	Frequency-Based Schedulerのアクセス権を変更
fbsattach	7-8	Frequency-Based Schedulerにタイミング・ソースを接続
fbsconfigure	7-9	Frequency-Based Schedulerを構成
fbscycle	7-11	マイナー・サイクル/メジャー・フレームの総数を返答
fbsdetach	7-12	Frequency-Based Schedulerからタイミング・ソースを分離
fbsgetrtc	7-13	リアルタイム・クロックの値を取得
fbsid	7-14	キーに対するFrequency-Based Scheduler識別子を返答
fbsinfo	7-15	Frequency-Based Schedulerの情報を返答
fbsintrpt	7-17	スケジューリングを開始/停止/再開
fbsquery	7-18	Frequency-Based Schedulerのプロセスを問合せ
fbsremove	7-21	Frequency-Based Schedulerを削除
fbsresume	7-22	Frequency-Based Schedulerのスケジューリングを再開

(次ページに続く)

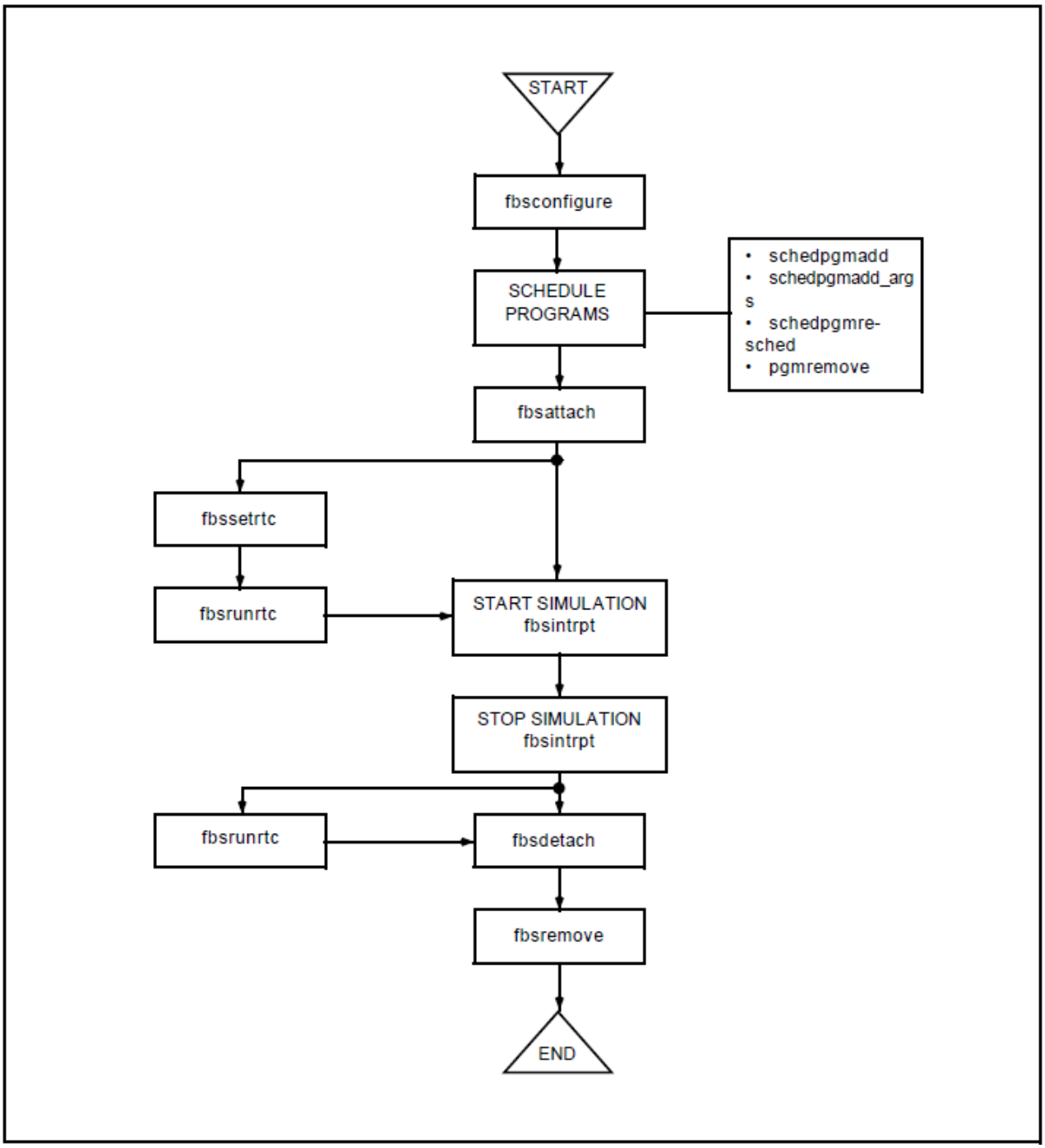
表7-1 Frequency-Based SchedulerのFORTRANライブラリ・ルーチン (続き)

ルーチン	ページ	説明
fbsrunrtc	7-24	リアルタイム・クロックを開始/停止
fbsschedself	7-25	Frequency-Based Schedulerに呼び出し元プロセスを追加
fbssetrtc	7-27	リアルタイム・クロックを設定
fbswait	7-28	Frequency-Based Schedulerを待機
nametopid	7-29	プロセスIDを返答
pgmquery	7-30	Frequency-Based Schedulerのプロセスを問合せ
pgmremove	7-32	Frequency-Based Schedulerからプロセスを削除
pgmreschedule	7-34	プロセスを再スケジュール
pgmschedule	7-37	Frequency-Based Schedulerにプロセスをスケジュール
pgmstat	7-40	FBSでスケジュールされたプロセスの状態を問合せ
pgmtrigger	7-42	Frequency-Based Schedulerのプロセスを起動
rtparm	7-43	初期パラメータを返答
sched_pgm_deadline_query	7-44	プロセスに割り当て済みデッドラインを問合せ
sched_pgm_deadline_test	7-46	デッドライン違反の有無をテスト
sched_pgm_set_deadline	7-48	プロセス・デッドライン時間を設定または消去
sched_pgm_set_soft_overrun_limit	7-50	ソフト・オーバーラン上限を設定
sched_pgm_soft_overrun_query	7-51	ソフト・オーバーランの処理を問合せ
schedfbsqry	7-52	Frequency-Based Schedulerのプロセスを問合せ
schedpgmadd	7-55	Frequency-Based Schedulerにプロセスをスケジュール
schedpgmadd_args	7-57	Frequency-Based Schedulerに引数を付けてプロセスをスケジュール
schedpgmqry	7-59	プロセスを問合せ
schedpgmresched	7-62	プロセスを再スケジュール

FORTRANライブラリ・コールの順序

アプリケーション・プログラムからルーチンを呼び出す可能性のあるおおよその順番を図 6-1 に示します。

図7-1 FORTRANライブラリ・コールの順序：Frequency-Based Scheduler



Frequency-Based Schedulerの使用

これ以降は、**libF77rt** ライブラリ内に含まれる Frequency-Based Scheduler ルーチンをアルファベット順で示します。

fbsaccess - Frequency-Based Schedulerのアクセス権を変更

本サブルーチンは選択した Frequency-Based Scheduler に割り当てられたアクセス権を変更します。アクセス権は Frequency-Based Scheduler の所有者/生成者と同じ実効ユーザーID を持つプロセスによってのみ変更することが可能です。

呼出し命令

```
call fbsaccess (schdle, uid, gid, permissions, istat)
integer schdle, uid, gid, permissions, istat;
```

パラメータ

schdle Frequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

uid 指定したFrequency-Based Schedulerの実効ユーザーIDを表す整数値。

gid 指定したFrequency-Based Schedulerの実効グループIDを表す整数値。

permissions 指定したFrequency-Based Schedulerに関連付けられたアクセス権を設定するために使用されるビット・パターン。ビット・パターンと対応するアクセス権を表7-2に示します。

表7-2 Frequency-Based Schedulerのアクセス権

ビット・パターン	アクセス権
400	ユーザーによる読み取り
200	ユーザーによる変更
060	グループによる読み取り、変更
006	その他ユーザーによる読み取り、変更

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。

返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**fbsaccess(3f)**のmanページを参照して下さい。

fbsattach - Frequency-Based Schedulerにタイミング・ソースを接続

本サブルーチンはタイミング・ソースを Frequency-Based Scheduler に接続、またはエンド・オブ・サイクル・スケジューリングを指定するために呼び出されます。タイミング・ソースはリアルタイム・クロックまたはエッジ・トリガー割込みデバイスにすることが可能です。

呼出し命令

```
call fbsattach (schdle, cpu, devname, istat)
integer schdle, cpu, istat;
character* (*) devname;
```

パラメータ

schdle タイミング・ソースが接続されている、もしくは指定したエンド・オブ・サイクル・スケジューリングのFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

cpu 値0を含む必要のある変数。

devname NULL文字列または指定したスケジューラーのタイミング・ソースとして使用するデバイスのパス名称。*devname*がNULL文字列である場合、エンド・オブ・サイクル・スケジューリングが指定されます。つまり、現在のマイナー・サイクルで実行するようにスケジュールされた最後のプロセスがそのサイクルで実行を終了した時、次のマイナー・サイクルでのプロセスの実行が発生します。

*devname*がパス名称を含む場合、リアルタイム・クロックまたはエッジ・トリガー割込みを参照することが可能です。タイミング・ソースのデバイス・ファイルに関する情報については3章を参照して下さい。

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**fbsattach(3f)**のmanページを参照して下さい。

fbsconfigure - Frequency-Based Schedulerを構成

本サブルーチンは Frequency-Based Scheduler を構成、または既に構成されている Frequency-Based Scheduler の構成詳細を取得するために呼び出されます。スケジューラーを構成するには、呼び出し元プロセスは *fbscheduser* ロール(特権に関する追加情報については 1 章を参照して下さい)に関連付けられたアクセス権を持っている必要があります。

スケジューラーを構成したい場合、Frequency-Based Scheduler に対してユーザーが選択した数値の識別子である *key* を指定する必要があります。*cycles* の値は以下を実行する機能を示します：

- *cycles*に指定した値が0ではない、かつ既存のFrequency-Based Schedulerの数がシステムに課せられた上限(デフォルト=10)未満である場合、Frequency-Based Schedulerが生成され識別子が返されます。
- *cycles*の値が0である場合、Frequency-Based Schedulerの現在の構成が返されます。

呼出し命令

```
call fbsconfigure （key, cycles, progs, max, reset, configflg, schdle, istat）
integer  key, cycles, progs, max, reset, configflg, schdle, istat
```

パラメータ

Frequency-Based Scheduler を生成するには、説明に従い以下のパラメータを指定する必要があります。

key	生成されるFrequency-Based Schedulerを特定する整数値。
cycles	指定したスケジューラーのフレームを構成するマイナー・サイクルの数を示す整数値。
progs	1マイナー・サイクル中にスケジュールして実行することが可能なプログラムの最大数を表す整数値。
max	指定したスケジューラーで同時にスケジュール可能なプログラムの最大数を表す整数値。この値は*cycles*および*progs*パラメータで指定した値を掛けて得られる積以下である必要があります。
reset	指定したスケジューラーに現在スケジュールされているプロセスをスケジューラーが再構成される前に終了するかどうかを表す整数値。許容値および対応する結果は次のとおりです： <0　指定したスケジューラーに現在スケジュールされている全てのプロセスを終了して削除 0　指定したスケジューラーに現在スケジュールされている全てのプロセスを無視 >0　指定したスケジューラーに現在スケジュールされている全てのプロセスを削除

configflg 指定したスケジューラーに割り当てられるアクセス権を表す整数値。

fbs_id Frequency-Based Schedulerの識別子を表す一意の正の整数値。この識別子はFBSとPerformance Monitorの殆どのライブラリ・サブルーチンで必要になることに注意することが重要です。

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**`fbsconfigure(3f)`**のmanページを参照して下さい。

既存の Frequency-Based Scheduler に関する情報を取得するには、説明に従い以下のパラメータを指定する必要があります。

key 構成情報が返されるFrequency-Based Schedulerを特定する整数値。この値がゼロである場合、このスケジューラーに関連付けられたFrequency-Based Scheduler識別子は*schdle*パラメータを使って提供される必要があります。

cycles 指定したスケジューラーの現在の構成情報が返されることを表す整数値はゼロ。**`fbsconfigure`**は本変数に指定したスケジューラーのフレームを構成するマイナー・サイクルの数を表す整数値を返します。

progs 指定したスケジューラーで1マイナー・サイクル中にスケジュールして実行することが可能なプログラムの最大数。

max 指定したスケジューラーで同時にスケジュール可能なプログラムの最大数。

configflg 指定したスケジューラーに割り当てられたアクセス権。

schdle 指定したFrequency-Based Schedulerの識別子を表す一意の正の整数値。*key*に**`0`**を指定した場合、本変数は関連するFrequency-Based Schedulerの識別子を含める必要があります。

fbscycle - マイナー・サイクル/メジャー・フレームの総数を返答

本サブルーチンは Frequency-Based Scheduler の現在のマイナー・サイクルおよびメジャー・フレームの総数値を取得するために呼び出されます。これらの値でシミュレーションの進捗を判定することが可能です。

呼出し命令

```
call fbscycle (schdle, count, istat)
integer schdle, count(2), istat
```

パラメータ

schdle 現在のサイクルとフレームの総数を取得したいFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

count 指定したスケジューラーの現在のマイナー・サイクルとメジャー・フレームを示す整数値を含む配列。*count(1)*は現在のマイナー・サイクルを含みます。*count(2)*はメジャー・フレームの総数を含みます。

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**fbscycle(3f)**のmanページを参照して下さい。

fbsdetach - Frequency-Based Schedulerからタイミング・ソースを分離

本サブルーチンは現在接続されているタイミング・ソースを Frequency-Based Scheduler から分離、またはエンド・オブ・サイクル・スケジューリングを無効にするために呼び出されます。タイミング・ソースがリアルタイム・クロックである場合、本サブルーチンを呼び出す前にそのクロックを停止することを推奨します。**fbsrunrtc** (本サブルーチンの説明は 7-24 ページを参照)の呼び出しをすることでそうすることが可能です。

呼出し命令

```
call fbsdetach (schdle, istat)
integer schdle, istat
```

パラメータ

schdle 現在接続されているタイミング・ソースを分離したい、またはエンド・オブ・サイクル・スケジューリングを無効にしたいFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**fbsdetach(3f)**のmanページを参照して下さい。

fbsgetrtc - リアルタイム・クロックの現在値を取得

本サブルーチンは Frequency-Based Scheduler に接続されたリアルタイム・クロックに関する現在のカウントおよび分解能を取得するために呼び出されます。

呼出し命令

```
call fbsgetrtc (schdle, count, resolution, istat1, istat2)
integer schdle, count, resolution, istat1, istat2
```

パラメータ

パラメータは示された順番に指定する必要があります。それらを以下で説明します。

schdle リアルタイム・クロックが接続されているFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

count マイナー・サイクルあたりの現在のクロック・カウント数を示す整数値。この値は1から65535の範囲となります。

resolution 現在の1クロック・カウントの持続時間をマイクロ秒単位で示す整数値。この値は1, 10, 100, 1000, 10000のいずれかです。

istat1 エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**fbsgetrtc(3f)**のmanページを参照して下さい。**open**または**ioctl**の呼び出しでエラーが発生したことを示す値が*istat1*に含まれている場合、この呼び出しのエラー・ステータスは*istat2*に返されます。

istat2 **open**または**ioctl**の呼び出しのエラー・ステータス。エラーの説明についてはインクルード・ファイル**<errno.h>**を参照して下さい。

fbsid - キーに対するFBS識別子を返答

本サブルーチンは固有のユーザー指定キーに関連付けられた Frequency-Based Scheduler 識別子を取得するために呼び出されます。このキーは **fbsconfigure(3f)** の呼び出しを行ってスケジューラーが生成された時に指定したキーと一致する必要があります。

呼出し命令

```
call fbsid (key, schdle, istat)
integer key, schdle, istat
```

パラメータ

パラメータは示された順番に指定する必要があります。それらを以下で説明します。

key	Frequency-Based Schedulerを特定する整数値。この値は **fbsconfigure** (本サブルーチンの説明は7-9ページ参照)の呼び出しを行ってスケジューラーが生成された時に*key*に対して指定した値と同じである必要があります。
schdle	*key*に関連付けられた一意のFrequency-Based Schedulerの識別子を表す整数値。
istat	エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**fbsid(3f)**のmanページを参照して下さい。

fbsinfo - Frequency-Based Schedulerの情報を返答

これらのルーチンは他のルーチンの呼び出しでは取得できない Frequency-Based Scheduler に関連する情報を取得します(例えば、**schedfbsqry**, **schedpgmqry**)。情報には以下が含まれます：

- スケジューラーの所有者および作成者のユーザーIDとグループID
- スケジューラーに割り当てられたアクセス権
- スケジューラーの識別子に関連付けられたキー
- スケジューラー上の全プロセスの合計オーバーラン数
- システムでアクティブなCPU
- Performance Monitorが有効になっているCPU
- FBS有効フラグ
- スケジューラーに接続されたデバイスのパス名称

呼出し命令

```
call fbsinfo (schdle, buf, devname, istat)
integer schdle, buf(41), istat
character* (*) devname
```

パラメータ

パラメータを以下で説明します。

schdle Frequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

buf 指定したスケジューラに関する情報を含む配列。配列の各エレメントに返される情報を表7-3に示します。

表7-3 配列エレメントの中身: fbsinfo

エレメント	中身
buf(1)	所有者のユーザーID
buf(2)	所有者のグループID
buf(3)	作成者のユーザーID
buf(4)	作成者のユーザーID
buf(5)	アクセス・モード
buf(6)	キー
buf(7)	フラグ・ワード
buf(8)	将来用に予約済み

表7-3 配列エレメントの中身: fbsinfo (続き)

エレメント	中身
buf(9)	スケジューラー上の全てのプロセスに関するハード・オーバーランの合計
buf(10)	システムでアクティブなCPUのマスク
buf(11)	Performance Monitorが有効になっているCPUのマスク
buf(12)	FBS有効フラグ
buf(13)-(41)	将来用に予約済み

devname 指定したFrequency-Based Schedulerのタイミング・ソースとして使用しているデバイスのパス名称。エンド・オブ・サイクル・スケジューリングが指定されている場合、*devname*はNULL文字列が含まれます。

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**`fbsinfo(3f)`**のmanページを参照して下さい。

fbsintrpt - Frequency-Based Schedulerのスケジューリングを開始/停止/再開

本サブルーチンは Frequency-Based Scheduler のスケジューリングを開始、停止、再開するために呼び出されます。スケジューリングを開始するために本サブルーチンを呼び出した場合、マイナー・サイクル、メジャー・フレーム、オーバーラン総数値はリセットされます。スケジューリングを再開するために呼び出した場合、それらの値はリセットされません。

fbsintrpt を呼び出す前にエンド・オブ・サイクル・スケジューリングを指定、またはスケジューリングを開始する Frequency-Based Scheduler にタイミング・ソースを接続するために **fbsattach** を起動する必要があります(**fbsattach** の説明は 7-8 ページを参照)。タイミング・ソースとしてリアルタイム・クロックを指定した場合、クロックを設定し開始するまでスケジューリングは始まりません(**fbssetrtc** と **fbsrunrtc** の説明はそれぞれ 7-27 ページと 7-24 ページを参照)。タイミング・ソースとしてエッジ・トリガー割込みデバイスを指定した場合、正常にスケジューリングを開始するため割り込みを既に生成している必要があります。

呼出し命令

```
call fbsintrpt （schdle, intrflag, istat）
integer schdle, intrflag, istat
```

パラメータ

schdle プロセスのスケジューリングを開始、停止、再開したいFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure (3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

intrflag 指定したスケジューラー上のプロセスのスケジューリングを開始、停止、または再開するのかを示す整数値。許容値および対応する結果は次のとおりとなります：

- <0 最初のフレーム、サイクル、オーバーラン総数の値をゼロに設定してプロセスのスケジューリングを開始します
- 0 プロセスのスケジューリングを停止し、現在のフレームとサイクルの総数を保存します
- >0 スケジューラーを最後に停止した時に保存された値をフレーム、サイクル、オーバーラン総数の値に設定してプロセスのスケジューリングを再開します

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**fbsintrpt(3f)**のmanページを参照して下さい。

fbsquery - Frequency-Based Schedulerのプロセスを問合せ

CAUTION

本インターフェースは廃止されました。これはCX/UXとの互換性のために維持していますが、関連付けれらたスケジューリング・ポリシーを示さずにプロセスのスケジューリング優先度を返します。このインターフェースを使用する既存のアプリケーションがある場合、**schedfbsqry(3f)** (7-52 ページを参照)を使用するようにそのアプリケーションを変更することを推奨します。

本サブルーチンはFrequency-Based Schedulerにスケジュールされたプロセスに関する情報を取得するために呼び出されます。ユーザーが指定したプロセッサーにスケジュールされた全てのプロセスに関する情報が返されます。各プロセスについて提供される情報は以下を含みます：

- プロセスを実行することが可能なCPUのマスク
- Frequency-Based SchedulerプロセスID
- スケジューリング優先度
- 周期(プロセスが各メジャー・フレームで起動される頻度を示すマイナー・サイクルの数)
- 開始ベース・サイクル(プロセスが各メジャー・フレームで起動される予定の最初のマイナー・サイクル)
- 「halt on overrun」フラグの値

呼出し命令

```
call fbsquery （schdle, cpu, buf1size, buf1, maxsize, buf2size, buf2, istat）
integer schdle, cpu, buf1size, buf1(buf1size), maxsize, buf2size, istat
character* （*） buf2
```

パラメータ

schdle スケジューリング情報を取得したいFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

cpu スケジューリング情報を取得するプロセッサーを示す整数値。許容値および対応する結果は次のとおりとなります：

0	呼び出しが行われたプロセッサー上で実行中のプロセスに関するスケジューリング情報が返されます
-1	スケジューラー上の全てのプロセスに関するスケジューリング情報が返されます

Bit mask (*cpu* & (1 << *i*))が設定された場合(*i*はゼロから15の範囲の整数でCPUを表します)、CPU *i*で実行中のプロセスに関するスケジューリング情報が返されます

buf1size *buf1*で表される配列のサイズを32bitワード単位で示す整数値。10ワードの情報がプロセスごとに返されるので、この値は10の倍数とすることを推奨します。

buf1 *cpu*パラメータで指定したプロセッサー上の各プロセスの連続する10個の整数値を含む配列。これらの値が返されるプロセスの数は*buf1size*パラメータの値に制約されます。例えば、*buf1size*の値が145である場合、14プロセスの値が返されます。これらの値はプロセスのスケジューリング情報を表します。単一プロセスにおける各配列エレメントに返される情報の種類を表7-4に示します。

表7-4 配列エレメントの中身: fbsquery

エレメント	中身
1	*buf2*内のプロセスのパス名称のバイト・オフセット
2	プロセスのパス名称のバイト長
3	ゼロ
4	ゼロ
5	プロセスが実行可能なCPUのマスク
6	プロセスのFrequency-Based SchedulerプロセスID
7	プロセスのスケジューリング優先度
8	プロセスが各メジャー・フレームで起動される頻度(期間)を示すマイナー・サイクルの数
9	プロセスが各メジャー・フレームで起動される予定の最初のマイナー・サイクル(開始ベース・サイクル)
10	「halt on overrun」フラグの値。ゼロ以外の値はフラグが設定されたことを示します。値ゼロはフラグが設定されていないことを示します。

maxsize *buf2*に返されるパス名称の最大長を示す整数値

buf2size *buf2*で表される文字列のサイズをバイト単位で示す整数値。*buf2*が取得したい全てのプロセスの名称を収容するのに十分な大きさであることを確実にするには、スケジューラーで許可されたプロセスの最大数(7-9ページに示された**`fbsconfigure`**の情報を参照して下さい)に32を掛けることで必要となるバイト数を計算するのに有用であると気付くかもしれません。

buf2	*cpu*パラメータで指定したプロセッサー上の各プロセスのパス名称。パス名称は一続きの文字列として返されます。各文字列の長さは*maxsize*の値以下です。フル・パス名称を収容するのに十分な大きさではない場合、途中で終わったコンポーネント名称が返されます。返されるパス名称の数は*buf2size*パラメータの値に制約されます。
istat	エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**`fbsquery(3f)`**のmanページを参照して下さい。

fbsremove - Frequency-Based Schedulerを削除

本サブルーチンは Frequency-Based Scheduler を削除してそれに関連付けられたデータ構造体を解放するために呼び出されます。**fbsremove** を呼び出す前にタイミング・ソースがスケジューラーから分離、もしくはエンド・オブ・サイクル・スケジューリングが無効(**fbsdetach** の使用に関する情報は 7-32 ページを参照)になっていることを確認する必要があることに注意することが重要です。**fbsremove** は指定したスケジューラーにスケジュールされている全てのプロセスを削除することに注意することが重要です。**fbsremove** を呼び出す前にスケジュールされている全てのプロセスを削除することを推奨します。**pgmremove** の呼び出し(本サブルーチンの使用に関する情報は 7-32 ページを参照)を行う事でそうすることが可能です。

Frequency-Based Scheduler を削除するには、呼び出し元プロセスはスケジューラーの所有者/作成者の実行ユーザーID と同じ実行ユーザーID を持っている必要があることに注意して下さい。

呼出し命令

```
call fbsremove （schdle, ab, istat）
integer schdle, ab, istat
```

パラメータ

schdle 削除したいFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

ab スケジューラーにスケジュールされているプロセスを処理する方法を示す整数値。許容値および対応する結果は次のとおり：

- <0 指定したスケジューラーに現在スケジュールされている全てのプロセスを終了して削除します
- ≧0 指定したスケジューラーに現在スケジュールされている全てのプロセスを削除します

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**fbsremove(3f)**のmanページを参照して下さい。

fbsresume - Frequency-Based Schedulerのスケジュールを再開

fbsresume サブルーチンは指定したマイナー・サイクル、メジャー・フレーム、オーバーラン総数で Frequency-Based Scheduler のプロセスのスケジューリングを再開するために呼び出されます。

Frequency-Based Scheduler のプロセスのスケジューリングを再開するには、呼び出し元プロセスがスケジューラへの変更許可を持っている必要があります。

スケジューラーの現在のフレーム、サイクル、オーバーランの値を変更せずに Frequency-Based Scheduler のプロセスのスケジューリングを再開したい場合、**fbsintrpt(3f)** サブルーチン(本サブルーチンの説明は 6-22 ページを参照)を使用することを推奨します。

CAUTION

fbsresume サブルーチンは指定したスケジューラーにスケジュールされている全てのプロセスに関する Performance Monitor の値を消去します。スケジューラーのフレームおよびサイクルの総数を変更すると Performance Monitor で保持している値が不正確となります。

呼出し命令

```
call fbsresume (schdle, frame, cycle, overruns, istat)
integer schdle, frame, cycle, overruns, istat
```

パラメータ

schdle	プロセスのスケジューリングを再開したいFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。
frame	指定したスケジューラーでプロセスのスケジューリングを再開したいメジャー・フレームを示す整数値。
cycle	指定したスケジューラーでプロセスのスケジューリングを再開したいマイナー・サイクルを示す整数値。この値はゼロからフレームあたりのマイナー・サイクルの総数-1の範囲とすることが可能です。フレームあたりのマイナー・サイクルの総数は、**fbsconfigure** (7-9ページ参照)を呼び出しを行ってスケジューラーが生成された時に指定しています。
overruns	指定したスケジューラーでスケジューリングを再開する時にオーバーラン総数に設定したい値を示す整数値。オーバーラン総数を変更したくない場合、値は-1を指定することが可能です。

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**fbsresume(3f)**のmanページを参照して下さい。

fbsrunrtc - リアルタイム・クロックを開始/停止

本サブルーチンは Frequency-Based Scheduler に接続されているリアルタイム・クロックのカウントを開始または停止するために呼び出されます。

呼出し命令

```
call fbsrunrtc (schdle, runflag, istat1, istat2)
integer schdle, runflag, istat1, istat2
```

パラメータ

schdle	接続されたリアルタイム・クロックを開始または停止したいFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。
runflag	リアルタイム・クロックを開始するのか停止するのかを示す整数値。ゼロ以外の値はクロックを開始することを示します。値ゼロはクロックを停止することを示します。
istat1	エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**fbsrunrtc(3f)**のmanページを参照して下さい。*istat1*に**open**または**ioctl**呼び出しに関するエラーが発生したことを示す値が含まれている場合、その呼び出しのエラー状況が*istat2*に返されます。
istat2	**open**または**ioctl**呼び出しのエラー状況。エラーの説明についてはインクルード・ファイル**<errno.h>**を参照して下さい。

fbsschedself - Frequency-Based Schedulerにプロセス/スレッドをスケジュール

fbsschedself サブルーチンは Frequency-Based Scheduler に呼び出し元プロセスまたはスレッドをスケジュールするために呼び出されます。

fbsschedself はプロセスにそのスケジューリング・ポリシーや優先度またはその CPU バイアスの設定を許可していないことに注意することが重要です。これらの作業は **fbsschedself** を呼び出す前に実行する必要があります。

プロセスは **sched_setscheduler(2)** ライブラリ・ルーチンを使用してそのスケジューリング・ポリシーと優先度を設定することが可能です。**mpadvise(3)** ライブラリ・ルーチンまたは **run(1)** コマンドを使用してその CPU バイアスを設定することが可能です。それらの機能を使用するための手順は *RedHawk Linux User's Guide* で説明されています。

Frequency-Based Scheduler に **/idle** を追加するために本ルーチンを使用できないことに注意して下さい。

Frequency-Based Scheduler に呼び出し元プロセスをスケジュールするには、それはスケジューラーに対する変更許可を持っている必要があります。

sched_setscheduler またはスケジューリング・ポリシーや優先度を変更することが可能な他のインターフェースを使ってスケジューラーにスケジュールされている間は、プロセスのスケジューリング・ポリシーまたは優先度を変更してはいけません。Frequency-Based Scheduler はこれらのインターフェースを使って行われたスケジューリング・ポリシーや優先度の変更を承知していません。

FBS でスケジュールされたシングル・スレッドのプロセスのスケジューリング・ポリシーまたは優先度を変更する必要がある場合、**schedpgmresched**(本サブルーチンの説明は 7-62 ページを参照)を使用することで実現可能です。

呼出し命令

```
call fbsschedself (schdle, name, sched_buf, istat)
integer schdle, istat
character* (*) name
integer (*) sched_buf
```

パラメータ

schdle	Frequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。
name	呼び出し元プログラムに関連付けられたプログラムを特定する標準パス名称または任意の情報。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。
sched_buf	スケジュールしたいプロセスのスケジューリング・パラメータを含む整数の配列。この配列に指定する情報を表7-5に示します。
istat	エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**fbsschedself(3f)**のmanページを参照して下さい。

表7-5 配列エレメントの中身: fbsschedself

エレメント	中身
sched_buf(1)	**fbsschedself**に渡される*sched_buf*のバージョンを示す整数値。この目的用に**<fbsched.h>**に定義されている記号定数**FBSSCHED_BUF_V1**を指定して下さい。
sched_buf(2)	FBSでスケジュールされるプロセスに渡される整数値。この値は**rtparm** (本サブルーチンの説明は7-43ページを参照)の呼び出しを通してFBSにスケジュールされたプロセスにより取得することが可能です。
sched_buf(3)	呼び出し元プロセスが各メジャー・フレームで起動される頻度を示す整数値。頻度1は呼び出し元プロセスがマイナー・サイクル毎に起動されることを示し、頻度2はマイナー・サイクル2回毎に1回起動されることを示します。 この値は1から**fbsconfigure** (本サブルーチンの説明は7-9ページを参照)の呼び出しで定義した時に指定したスケジューラーのフレームを構成するマイナー・サイクルの数の範囲とすることが可能です。
sched_buf(4)	呼び出し元プロセスが各フレームで起動するようにスケジュールされている最初のマイナー・サイクルを示す整数値。この値はゼロからフレームあたりのマイナー・サイクルの総数-1の範囲とすることが可能です。フレームあたりのマイナー・サイクルの総数は、**fbsconfigure** (本サブルーチンの説明は7-9ページを参照)の呼び出しで指定されています。
sched_buf(5)	呼び出し元プロセスでフレーム・オーバーランが発生した場合にスケジューラーを停止させるべきかどうかを示す整数値。ゼロ以外の値はスケジューラーを停止することを示します。
sched_buf(6)	**fbsschedself**が返す整数値で、これはスケジュールされるプロセスに関する一意のFrequency-Based SchedulerプロセスIDです。

fbssetrtc - リアルタイム・クロックを設定

本サブルーチンはリアルタイム・クロックのカウントと分解能の値を設定してマイナー・サイクルの持続時間を規定するために呼び出されます。

呼出し命令

```
call fbssetrtc (schdle, count, resolution, istat1, istat2)
integer schdle, count, resolution, istat1, istat2
```

パラメータ

schdle リアルタイム・クロックが接続されたFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

count マイナー・サイクルあたりのクロック・カウントの数を示す整数値。この値は2から65535の範囲が可能です。

resolution 1クロック・カウントのマイクロ秒単位での持続期間を示す整数値。この値は次のいずれかにする必要があります：1, 10, 100, 1000, 10000。

istat1 エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**fbssetrtc(3f)**のmanページを参照して下さい。*istat1*に**open**または**ioctl**呼び出しに関するエラーが発生したことを示す値が含まれている場合、その呼び出しのエラー状況が*istat2*に返されます。

istat2 **open**または**ioctl**呼び出しのエラー状況。エラーの説明についてはインクルード・ファイル**<errno.h>**を参照して下さい。

注意事項： *count* に 1 を使用することは出来ませんが、*resolution* の値と等しいタイミング間隔は、次に低い *resolution* の値と *count* に 10 を指定することにより設定することが可能となります。例えば、*resolution* が 1000 および *count* が 10 で 10,000 マイクロ秒のタイミング間隔をもたらします。

fbswait - Frequency-Based Schedulerを待機

本サブルーチンは Frequency-Based Scheduler にスケジュールされているプロセスを次にスケジュールされているマイナー・サイクルまでスリープすることを可能にします。

スケジュールされているプロセスが次にスケジュールされているマイナー・サイクルまでに本ライブラリ・ルーチンをコールしない場合、ソフト・オーバーランまたはハード・オーバーランのどちらかが発生します。

スケジュールされているマイナー・サイクルを連続的に逃したプロセス毎の総数がプロセス毎のソフト・オーバーラン上限に達していないまたは超えていない場合、ソフト・オーバーランは発生します。ソフト・オーバーランが発生した時、次にスケジュールされているマイナー・サイクルを待つためのブロックの代わりにプロセスは **fbswait** 呼び出しからすぐに戻ります。

スケジュールされているマイナー・サイクルを連続的に逃した総数がプロセス毎のソフト・オーバーラン上限に達したまたは超えた場合、ハード・オーバーランが発生します。この場合、プロセスは次にスケジュールされているマイナー・サイクルまで **fbswait** でブロックします。

プロセスの連続的なソフト・オーバーラン上限は、**sched_pgm_set_soft_overrun_limit(3f)** を使って既定値の 0 から変更することが可能です。**pmqrypgm(3f)** を使って読み取り可能なハード・オーバーラン総数および **sched_pgm_soft_overrun_query(3f)** を介して読み取り可能なソフト・オーバーラン総数はプロセスが割り当てられた頻度で実際に動作しているかどうかを示します。

スケジュールされているプロセスがデッドラインの影響下にあり、スケジュールされているプロセスがそのデッドライン時間を過ぎた後にこのサービスを呼び出す場合、デッドライン違反を検出してスケジューラーは停止される可能性があります。

呼出し命令

```
call fbswait (istat)
integer istat
```

パラメータ

istat エラーが発生したかどうか、スケジューラーもしくは他のプロセスからの**pgmtrigger(3f)**コールにより起動されたかどうかを示す整数値。返される可能性がある値は以下のとおりです：

0	プロセスは正常に起動された
1	プロセスは**pgmtrigger(3f)**コールの結果として起動された
2	カーネルはソフト・オーバーランを検出してプロセスにそこからの回復を試みさせるため、プロセスはスリープしなかった
他の0以外の値	特定の種類のエラーが発生した。返される可能性のあるゼロ以外の値およびそれが表すエラーの種類を表示するには**fbswait(3f)**のmanページを参照して下さい。

nametopid - 指定したプロセス名称のプロセスIDを返答

nametopid ルーチンは指定したプロセスのプロセス ID (pid)を返します。

呼出し命令

```
call nametopid (name, key, cpu, pid, istat)
integer key, cpu, pid, istat
character* (*) name
```

パラメータ

name プロセスを特定するパス名称

key Frequency-Based Schedulerを特定する整数値。この値は **fbsconfigure** (本サブルーチンの説明は7-9ページを参照)の呼び出しを行ってスケジューラーが生成された時に*key*に対して指定した値と同じである必要があります。プロセスがFrequency-Based Schedulerにスケジュールされていないことを示すために-1を設定することも可能です。

cpu 情報が返されるプログラムを特定するために*name*パラメータの値と関連して使用されるプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
-1	いずれかのプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
Bit mask	(*cpu* & (1 << *i*))が設定され(*i*はゼロから15の範囲の整数でCPUを表します)、それが唯一のビット・セットである場合、CPU *i*で実行中の*name*で指定された最初のプロセスが指定されます
	(*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name*で指定された最初のプロセスが指定されます

pid *name*に指定したプロセスのプロセスID

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**nametopid(3f)**のmanページを参照して下さい。

pgmquery - Frequency-Based Schedulerのプロセスを問合せ

CAUTION

本インターフェースは廃止されました。**schedpgmqry(3f)** (7-59 ページを参照)を使用することを推奨します。

本サブルーチンが返す情報は以下を含みます：

- プロセスのパス名称
- プロセスを実行することが可能なCPU
- Frequency-Based SchedulerプロセスID
- スケジューリング優先度
- 周期(プロセスが各メジャー・フレームで起動される頻度を示すマイナー・サイクルの数)
- 開始ベース・サイクル(プロセスが各メジャー・フレームで起動される予定の最初のマイナー・サイクル)
- 「halt on overrun」フラグの値

呼出し命令

```
call pgmquery (schdle, name, cpu, slot, prior, period, cycle, ab, istat)
integer schdle, cpu, slot, prior, period, cycle, ab, istat
character* (*) name
```

パラメータ

schdle スケジューリング情報を取得したいプロセスがスケジュールされたFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

name 情報が返されるプロセスを特定する標準パス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。本変数が空白である場合、*slot*パラメータにFrequency-Based SchedulerプロセスIDを提供する必要があります。

cpu 情報が返されるプログラムを特定するために*name*パラメータの値と関連して使用されるプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
-1	いずれかのプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます

	Bit mask	(*cpu* & (1 << *i*))が設定され(*i*はゼロから15の範囲の整数でCPUを表します)、それが唯一のビット・セットである場合、CPU *i*で実行中の*name*で指定された最初のプロセスが指定されます (*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name*で指定された最初のプロセスが指定されます
slot		情報が返されるプロセスに関する一意のFrequency-Based Schedulerプロセスを提供する整数値。この値は**pgmschedule** (本サブルーチンの説明は7-37ページを参照)の呼び出しを行ったときに取得します。*name*と*cpu*を指定するだけで問い合わせるプログラムを特定したい場合、この値は-1にする必要があります。
prior		指定したプロセスのスケジューリング優先度を示す整数値
period		指定したプログラムが各メジャー・フレームで起動される頻度を示す整数値
cycle		指定したプロセスが各フレームで起動される予定の最初のマイナー・サイクルを示す整数値
ab		「halt on overrun」フラグの値を示す整数値。ゼロ以外の値はフラグが設定されることを示します。値が0はフラグが設定されないことを示します。
istat		エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**pgmquery(3f)**のmanページを参照して下さい。

pgmremove - Frequency-Based Schedulerからプロセスを削除

本サブルーチンは Frequency-Based Scheduler からプロセスを削除するために呼び出されます。以下の方法のいずれかを使って削除したプロセスを特定することが可能です：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based SchedulerプロセスID(スロット番号)を指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based SchedulerプロセスIDを指定

NOTE

同じ CPU 上に複数回スケジュールされたプロセスを特定することが可能な唯一の方法は、その Frequency-Based Scheduler プロセス ID を指定することです。

呼出し命令

```
call pgmremove (schdle, name, cpu, slot, ab, istat)
integer schdle, cpu, slot, ab, istat
character* (*) name
```

パラメータ

schdle プロセスがスケジュールされたFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

name 指定したスケジューラーから削除するプロセスを特定する標準パス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。本変数が空白である場合、*slot*パラメータにFrequency-Based SchedulerプロセスIDを提供する必要があります。

cpu 指定したスケジューラーから削除するプロセスを特定するために*name*パラメータの値と関連して使用されるプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが削除されます
-1	いずれかのプロセッサーで現在実行中の*name*で指定された最初のプロセスが削除されます
Bit mask	(*cpu* & (1 << *i*))が設定され(*i*はゼロから15の範囲の整数でCPUを表します)、それが唯一のビット・セット

である場合、CPU *i*で実行中の*name*で指定された最初のプロセスが指定されます

(*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name*で指定された最初のプロセスが指定されます

slot 指定したスケジューラーから削除されるプロセスに関する一意のFrequency-Based SchedulerプロセスIDを提供する整数値。この値は**schedpgmadd**(本サブルーチンの説明は7-55ページを参照)の呼び出しを行ったときに取得します。*name*と*cpu*を指定するだけで問い合わせるプログラムを特定したい場合、この値は-1にする必要があります。

ab 指定したプロセスが指定したスケジューラーから削除される方法を示す整数値。正の値はスケジューラーからプロセスは削除されますが実行を継続することを示します。負の値はプロセスはスケジューラーから削除され終了することを示します。

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**pgmremove(3f)**のmanページを参照して下さい。

pgmreschedule - プロセスを再スケジュール

CAUTION

本インターフェースは廃止されました。これはCX/UXとの互換性のために維持していますが、プロセスのスケジューリング優先度の仕様に対して動作が変更されました。このインターフェースを使用する既存のアプリケーションがある場合、**schedpgmresched(3f)** (7-62 ページを参照)を使用するようにそのアプリケーションを変更することを推奨します。

本サブルーチンはFrequency-Based Schedulerにスケジュールされているプロセスに関するスケジューリング・パラメータを変更するために呼び出されます。例えば、プログラムの優先度もしくは実行がスケジュールされている頻度を変更したくなる場合があります。しかしながら、スケジュールされたCPUを変更することは出来ません。

プロセスの優先度を変更するには、以下の条件が満たされる必要があります：

- 呼び出し元プロセスは*fbscheduser*ロールに関連付けられた機能を所有している必要があります(第1章を参照)
- 呼び出し元プロセスの実効ユーザーIDはターゲット・プロセス(スケジューリング・ポリシーと優先度が設定されているプロセス)の実効ユーザーIDと一致している、もしくは呼び出し元プロセスはP_OWNER特権を持っている必要があります。

pgmremove を呼び出してスケジューラーからプロセスを削除(7-32 ページを参照)もしくは **fbsintrpt** を呼び出してシミュレーションを停止(7-17 ページを参照)することなく、**pgmreschedule** を呼び出してパラメータを変更することが可能です。

以下のいずれかの方法を使って再スケジュールするプロセスを特定することが可能です：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based SchedulerプロセスID(スロット番号)を指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based SchedulerプロセスIDを指定

NOTE

同じCPU上で複数回スケジュールされているプロセスを識別するために使用することが可能な唯一の方法は、そのFrequency-Based Scheduler プロセスIDを指定することです。

呼出し命令

```
call pgmreschedule (schdle, name, cpu, slot, prior, param, period, cycle, ab, istat)
integer schdle, cpu, slot, prior, param, period, cycle, ab, istat
character* (*) name
```

パラメータ

schdle プロセスがスケジュールされているFrequency-Based Schedulerの識別子を表す一意の正の整数。**`fbsconfigure(3f)`** (本サブルーチンの説明は7-9ページを参照)または**`fbsid(3f)`** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

name 再スケジュールするプロセスを特定する標準パス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。本変数が空白である場合、*slot*パラメータにFrequency-Based SchedulerプロセスIDを提供する必要があります。

cpu 再スケジュールするプロセスを特定するために*name*パラメータの値と関連して使用されるプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが再スケジュールされます
-1	いずれかのプロセッサーで現在実行中の*name*で指定された最初のプロセスが再スケジュールされます
Bit mask	(*cpu* & (1 << *i*))が設定され(*i*はゼロから15の範囲の整数でCPUを表します)、それが唯一のビット・セットである場合、CPU *i*で実行中の*name*で指定された最初のプロセスが再スケジュールされます
	(*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name*で指定された最初のプロセスが再スケジュールされます

slot 再スケジュールするプロセスに関する一意のFrequency-Based SchedulerプロセスIDを提供する整数値。この値は**`pgmschedule`** (本サブルーチンの説明は7-37ページを参照)の呼び出しを行ったときに取得します。*name*と*cpu*を指定するだけで問い合わせるプログラムを特定したい場合、この値は-1にする必要があります。

prior 指定したプロセスのスケジューリング優先度を示す整数値。**`pgmschedule`** (本サブルーチンの説明は7-37ページを参照)を使ってスケジュールされたプロセスはPOSIXの**SCHED_RR**スケジューリング・ポリシーに従いスケジュールされます。指定した値は関連するポリシーの優先度の範囲内にある必要があります。優先度の許容範囲はシェルからオプションや引数を指定することなく**`run(1)`**コマンドを起動(本コマンドの説明については対応するmanページを参照して下さい)することで取得することが可能です。より高い数値はより有益なスケジューリング優先度に対応します。

スケジューリング・ポリシーと優先度に関する詳細については*RedHawk Linux User's Guide*を参照して下さい。

param — Frequency-Based Schedulerにスケジュールされるプロセスに渡される整数値

period — 指定したプログラムが各メジャー・フレームで起動される頻度を示す整数値。頻度1は指定したプログラムがマイナー・サイクル毎に起動されることを示し、頻度2はマイナー・サイクル2回毎に1回、頻度3はマイナー・サイクル3回毎に1回起動されることを示します。この値は1から**fbsconfigure** (7-9ページを参照)の呼び出しで定義した時に指定したスケジューラーのフレームを構成するマイナー・サイクルの数の範囲とすることが可能です。

cycle — 指定したプロセスが各フレームで起動するようにスケジュールされている最初のマイナー・サイクルを示す整数値。この値はゼロからフレームあたりのマイナー・サイクルの総数-1の範囲とすることが可能です。フレームあたりのマイナー・サイクルの総数は、**fbsconfigure** (本サブルーチンの説明は7-9ページを参照)の呼び出しで指定されています。

ab — 指定したプロセスでフレーム・オーバーランが発生した場合にスケジューラーを停止させるべきかどうかを示す整数値。ゼロ以外の値はスケジューラーを停止することを示します。

istat — エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**pgmreschedule(3f)**のmanページを参照して下さい。

pgmschedule - Frequency-Based Schedulerにプロセスをスケジュール

CAUTION

本インターフェースは廃止されました。これはCX/UXとの互換性のために維持していますが、プロセスのスケジューリング優先度の仕様に対して動作が変更されました。このインターフェースを使用する既存のアプリケーションがある場合、**`schedpgmadd(3f)`** (7-55 ページを参照)を使用するようにそのアプリケーションを変更することを推奨します。

本サブルーチンは新しいプロセスを生成しFrequency-Based Schedulerにそれをスケジュールするために呼び出されます。プロセスが本サブルーチンを使ってスケジュールされる場合、それはPOSIXの**SCHED_RR**スケジューリング・ポリシーに従いスケジュールされます(スケジューリング・ポリシーと優先度に関する詳細については*RedHawk Linux User's Guide*を参照して下さい)。

プロセスのスケジューリング優先度を設定する場合、以下の条件が満たされる必要があります：

- 呼び出し元プロセスは*fbscheduser*ロールに関連付けられた機能を所有している必要があります(第1章を参照)
- 呼び出し元プロセスの実効ユーザーIDはターゲット・プロセス(スケジューリング・ポリシーと優先度が設定されているプロセス)の実効ユーザーIDと一致している、もしくは呼び出し元プロセスはP_OWNER特権を持っている必要があります。

本サブルーチンを呼び出す時にプロセスのCPUバイアスを修正したい場合、呼び出し元プロセスの実/実効ユーザーIDはCPU割り当てを変更するプロセスの実ユーザーIDまたは保存されたユーザーIDと一致する必要があります。

呼出し命令

```
call pgmschedule (schdle, name, prior, param, period, cycle, ab, cpu, slot, istat)
integer schdle, prior, param, period, cycle, ab, cpu, slot, istat
character* (*) name
```

パラメータ

schdle — Frequency-Based Schedulerの識別子を表す一意の正の整数。**`fbsconfigure(3f)`** (本サブルーチンの説明は7-9ページを参照)または**`fbsid(3f)`** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

name — スケジューラーにスケジュールするプログラムを特定する標準パス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。

prior — 指定したプロセスのスケジューリング優先度を示す整数値。**`pgmschedule`**を使ってスケジュールされるプロセスはPOSIXの**SCHED_RR**スケジューリング・ポリシーに従いスケジュールされます。指定した値は関連するポリシーの優先度の範囲内にある必要があります。

優先度の許容範囲はシェルからオプションや引数を指定することなく**run(1)**コマンドを起動(本コマンドの説明については対応するmanページを参照して下さい)することで取得することが可能です。より高い数値はより有益なスケジューリング優先度に対応します。

スケジューリング・ポリシーと優先度に関する詳細については*RedHawk Linux User's Guide*を参照して下さい。

param Frequency-Based Schedulerにスケジュールされるプロセスに渡される整数値。この値は**rtparm**への呼び出し(本サブルーチンの説明は7-43ページを参照)を通してFBSでスケジュールされたプロセスで回収することが可能です。

period 指定したプログラムが各メジャー・フレームで起動される頻度を示す整数値。頻度1は指定したプログラムがマイナー・サイクル毎に起動されることを示し、頻度2はマイナー・サイクル2回毎に1回、頻度3はマイナー・サイクル3回毎に1回起動されることを示します。この値は1から**fbsconfigure** (7-9ページを参照)の呼び出しで定義した時に指定したスケジューラーのフレームを構成するマイナー・サイクルの数の範囲とすることが可能です。

cycle 指定したプロセスが各フレームで起動するようにスケジュールされている最初のマイナー・サイクルを示す整数値。この値はゼロからフレームあたりのマイナー・サイクルの総数-1の範囲とすることが可能です。フレームあたりのマイナー・サイクルの総数は、**fbsconfigure** (本サブルーチンの説明は7-9ページを参照)の呼び出しで指定されています。

ab 指定したプロセスでフレーム・オーバーランが発生した場合にスケジューラーを停止させるべきかどうかを示す整数値を含んでいるフラグを参照します。ゼロ以外の値はスケジューラーを停止することを示します。

cpu 指定したプログラムの実行をスケジュール可能なプロセッサーを識別するマスクを参照します。許容値および対応する結果は次のとおり：

0	*name*で指定されたプログラムは呼び出しが行われるプロセッサーにスケジュールすることが可能
-1	*name*で指定されたプログラムはいずれのプロセッサーでもスケジュールすることが可能
Bit mask	(*cpu* & (1 << *i*))が設定された場合(*i*はゼロから15の範囲の整数でCPUを表します)、*name*で指定されたプログラムはCPU *i*にスケジュールすることが可能

slot **pgmschedule**がスケジュールされたプロセスの一意のFrequency-Based SchedulerプロセスIDである整数値を返す変数を参照します。

istat **pgmschedule**がエラーが発生したかどうかを示す整数値を返す変数を参照します。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**pgmschedule(3f)**のmanページを参照して下さい。

pgmstat - FBSでスケジュールされたプロセスの状態を問合せ

本サブルーチンは Frequency-Based Scheduler にスケジュールされた特定のプロセスの状態に関する情報を取得するために呼び出されます。プロセスの状態はそれが **fbswait** のスリープ状態なのかそれ以外の状態なのかどうかを示します。

以下の方法のいずれかを使ってプロセスを特定することが可能です：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based Schedulerプロセス ID(スロット番号)を指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based Schedulerプロセス IDを指定

NOTE

同じ CPU 上に複数回スケジュールされたプロセスを特定することが可能な唯一の方法は、その Frequency-Based Scheduler プロセス ID を指定することです。

返される情報は以下を含みます：

- プロセスのパス名称
- プロセスを実行することが可能なCPUのマスク
- Frequency-Based SchedulerプロセスID
- プロセスの現在の状態

呼出し命令

```
call pgmstat (schdle, name, cpu, slot, state, istat)
integer schdle, cpu, slot, state, istat
character* (*) name
```

パラメータ

schdle 状態情報を取得したいプロセスがスケジュールされたFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure (3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

name 状態情報が返されるプロセスを特定する標準パス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。本変数が空白である場合、*slot*パラメータにFrequency-Based Schedulerプロセス IDを提供する必要があります。**pgmstat**はこの変数に指定したFBSでスケジュールされたプロセスのパス名称を返します。

cpu 情報が返されるプログラムを特定するために*name*パラメータの値と関連して使用されるプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
-1	いずれかのプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
Bit mask	(*cpu* & (1 << *i*))が設定され(*i*はゼロから15の範囲の整数でCPUを表します)、それが唯一のビット・セットである場合、CPU *i*で実行中の*name*で指定された最初のプロセスが指定されます
	(*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name*で指定された最初のプロセスが指定されます

pgmstatはこの変数に指定したプロセスが実行可能なCPUのマスクを返します。

slot 状態情報が返されるプロセスに関する一意のFrequency-Based SchedulerプロセスIDを提供する整数値。この値は**schedpgmadd** (本サブルーチンの説明は7-55ページを参照)の呼び出しを行ったときに取得します。*name*と*cpu*を指定するだけで問い合わせるプログラムを特定したい場合、この値は-1にする必要があります。**pgmstat**はこの変数に指定したプロセスのFrequency-Based SchedulerプロセスIDを返します。

state **<fbsched.h>**で定義されている指定したプロセスの現在の状態を示す整数値

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**pgmstat(3f)**のmanページを参照して下さい。

pgmtrigger - FBSの待ち状態プロセスを起動

本サブルーチンは **fbswait** でスリープ状態にあるプロセスを起こす処理を可能にします。呼び出し元プロセスは Frequency-Based Scheduler にスケジュールされている必要はなく、ターゲットのプロセスはそうである必要があることに注意することが重要です。

呼出し命令

```
call pgmtrigger (schdle, slot, tgrflg, istat)
integer schdle, slot, tgrflg, istat
```

パラメータ

schdle	スリープ中のプロセスがスケジュールされたFrequency-Based Schedulerの識別子を表す一意の正の整数。
slot	スリープ中のプロセスに関する一意のFrequency-Based Schedulerプロセスを提供する整数値。この値は**schedpgmadd** (本サブルーチンの説明は7-55ページを参照)の呼び出しを行ったときに取得します。
tgrflg	起動されたプロセスが実行しているプロセッサーでコンテキスト・スイッチを強制するかどうかを示す整数値。ゼロ以外の値はコンテキスト・スイッチが強制されることを示します。
istat	エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**pgmtrigger(3f)**のmanページを参照して下さい。

rtparm - 初期パラメータを返答

本サブルーチンは Frequency-Based Scheduler にスケジュールされたプロセスが **schedpgmadd** (7-55 を参照)または **schedpgmresched** (7-62 を参照)の呼び出しを介して渡されたプロセス初期パラメータの値を取得できるようにします。

呼出し命令

```
call rtparm (param)
integer param
```

パラメータ

param	**schedpgmadd**または**schedpgmresched**の呼び出しを介してプロセスに渡された整数値

sched_pgm_deadline_query - プロセスに割り当て済みデッドラインを問合せ

本サブルーチンは Frequency-Based Scheduler に現在スケジュールされているプロセスまたはスレッドに関するデッドラインのパラメータを問い合わせます。

この機能は新しいデッドライン違反を検出しないことに注意することが重要です。新しいデッドライン違反の検出を実行するには **sched_pgm_deadline_test(3f)** を使用して下さい。

プロセスは以下の方法のいずれかで識別することが可能です：

- スロットのみ(*name*が空白の場合)
- パス名称とプロセッサーIDのペアのみ(*slot*が-1の場合)
- スロットおよびパス名称とプロセッサーIDのペアの両方

呼出し命令

```
call sched_pgm_deadline_query (schdle, name, cpu, slot, ddln_halt, ddln_kind,
      ddln_origin, ddln_sec, ddln_nsec, violations, istat)
integer schdle, cpu, slot, ddln_halt, ddln_kind, ddln_origin, ddln_sec, ddln_nsec, violations, istat
character* (*) name
```

パラメータ

schdle — **fbsconfigure(3f)** または **fbsid(3f)** ライブラリ・ルーチンの呼び出しから取得するもしくは-1を設定。-1はFBSでスケジュールされたプロセスがスケジューラーの識別子を認識せずにそれがスケジュールされたFrequency-Based Schedulerを参照出来るようにします。

name — プロセス(またはマルチスレッド化したプロセスのスレッド)を特定するパス名称。名称が全て空白である場合、*slot*領域(Frequency-Based SchedulerプロセスID)が提供される必要があります。

cpu — ビット・マスクもしくは0または-1を設定のいずれか。ビット・マスクを指定した場合、(*cpu* & (1 << *i*))が設定されたプロセッサーが要求されます。*cpu*が0場合、呼び出しが行われるプロセッサーが要求されます。*cpu*が-1場合、全てのプロセッサーが要求されます。要求されたプロセッサーの1つで現在実行中の*name*で指定された最初のプロセスが返されます。

slot — プロセスのFrequency-Based SchedulerプロセスID。スロット番号が-1である場合、名称とプロセッサーIDが提供される必要があります。

ddln_halt — デッドライン違反の検出でスケジューラーは停止するかどうかを示します。この値は次のいずれかが可能です：

0 DL_NOHALT
: デッドライン違反の検出でスケジューラーは停止すべきではないことを示します。

1 DL_HALT
: デッドライン違反の検出でスケジューラーは停止すべきであることを示します。

ddln_kind
: **sched_pgm_set_deadline(3f)**の呼び出しでプロセスに設定されるデッドラインの種類。この値は*ddln_sec*と*ddln_nsec*の値の解釈に影響します。*ddln_kind*は次のいずれかが可能です：

  0 DEADLINE_CLEAR
  : 指定したプロセスまたはスレッドにデッドラインが現在割り当てられていないことを示します。

  1 DEADLINE_WALL_TIME
  : *ddln_sec*と*ddln_nsec*の値を同時にプロセスまたはスレッドに割り当てられたCLOCK_MONOTOICデッドライン時間に指定することを示します。

ddln_origin
: デッドライン時間が測定されるポイントを示します。これは次のいずれかが可能です：

  0 DL_CYCLE_RELATIVE
  : スケジュールされたタスクのサイクルの開始からデッドライン時間が測定されることを示します。

  1 DL_TASK_RELATIVE
  : スケジュールされたタスクが**fbswait(3f)**を終了して実行を開始する時間からデッドライン時間が測定されることを示します。

ddln_sec
ddln_nsec
: 指定した起点からプロセスが**fbswait(3f)**に戻ることを要求するまでに測定された最大時間をこれらは表します。

violations
: 指定したプロセスまたはタスクで発生したデッドライン違反の総数を示します。

istat
: エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するにはmanページを参照して下さい。

sched_pgm_deadline_test - デッドライン違反の有無をテスト

本サブルーチンは Frequency-Based Scheduler に現在スケジュールされているプロセスまたはスレッドによるデッドライン違反の発生についてテストします。

プロセスは以下の方法のいずれかで識別することが可能です：

- スロットのみ(*name*が空白の場合)
- パス名称とプロセッサーIDのペアのみ(*slot*が-1の場合)
- スロットおよびパス名称とプロセッサーIDのペアの両方

呼出し命令

```
call sched_pgm_deadline_test (schdle, name, cpu, slot, violated, violations, ddln_kind,
      ddln_origin, remaining_sec, remaining_nsec, istat)
integer  schdle, cpu, slot, violated, violations, ddln_kind, ddln_origin, remaining_sec,
      remaining_nsec, istat
character* (*) name
```

パラメータ

schdle — **fbsconfigure(3f)**または**fbsid(3f)**ライブラリ・ルーチンの呼び出しから取得するもしくは-1を設定。-1はFBSでスケジュールされたプロセスがスケジューラーの識別子を認識せずにそれがスケジュールされたFrequency-Based Schedulerを参照出来るようにします。

name — プロセスを特定するパス名称。名称が全て空白である場合、*slot*領域(Frequency-Based SchedulerプロセスID)が提供される必要があります。

cpu — ビット・マスクもしくは0または-1を設定のいずれか。ビット・マスクを指定した場合、(*cpu* & (1 << *i*))が設定されたプロセッサーが要求されます。*cpu*が0場合、呼び出しが行われるプロセッサーが要求されます。*cpu*が-1場合、全てのプロセッサーが要求されます。要求されたプロセッサーの1つで現在実行中の*name*で指定された最初のプロセスが返されます。

slot — プロセスのFrequency-Based SchedulerプロセスID。スロット番号が-1である場合、名称とプロセッサーIDが提供される必要があります。

violated — デッドライン違反の発生。これは次のいずれかが可能です：

0 NO_VIOLATION
: 指定したプロセスまたはスレッドでデッドライン違反が発生していないことを示します。

1 DEADLINE_VIOLATION
: デッドライン違反が発生したことを示します。

violations — スケジューラーが開始されてからのプロセスのデッドライン違反の数。

ddln_kind — **sched_pgm_set_deadline(3f)**の呼び出しでプロセスに設定されるデッドラインの種類。この値は*remaining_sec*と*remaining_nsec*の値の解釈に影響します。*ddln_kind*は次のいずれかが可能です：

0 DEADLINE_CLEAR
指定したプロセスまたはスレッドにデッドラインが現在割り当てられていないことを示します。

1 DEADLINE_WALL_TIME
*remaining_sec*と*remaining_nsec*の値を同時にデッドライン終了までのCLOCK_MONOTOICの残り時間に指定することを示します。

ddln_origin
`sched_pgm_set_deadline(3f)`呼び出しによってプロセスに設定されるデッドラインの起点を表します。この値は*remaining_sec*と*remaining_nsec*の値の解釈に影響します。*ddln_origin*は次のいずれかが可能です：

0 DL_CYCLE_RELATIVE
残り時間はサイクルの開始から測定されることを示します。

1 DL_TASK_RELATIVE
残り時間はタスク実行の開始から測定されることを示します。

remaining_sec
remaining_nsec
`fbswait(3)`に戻る前にプロセスが実行に費やすことが期待される残り時間をこれらは表します。

istat
エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するにはmanページを参照して下さい。

sched_pgm_set_deadline - プロセス・デッドライン時間を設定または消去

本サブルーチンは Frequency-Based Scheduler に現在スケジュールされているプロセスまたはスレッドのデッドライン時間を設定または消去します。

プロセスのデッドライン時間は**fbswait(3f)**に戻る前にプロセスが実行に費やすことが期待される最大時間を示します。デッドライン時間を超えた場合、デッドライン違反がプロセスで生じます。スケジューラーをオプションでデッドライン違反の検出で停止させることが可能です。

デッドラインを設定または消去するには、呼び出し元プロセスはスケジューラーに対する変更許可を持っている必要があります。

プロセスは以下の方法のいずれかで識別することが可能です：

- スロットのみ(*name*が空白の場合)
- パス名称とプロセッサーIDのペアのみ(*slot*が-1の場合)
- スロットおよびパス名称とプロセッサーIDのペアの両方

呼出し命令

```
call sched_pgm_set_deadline (schdle, name, cpu, slot, ddln_halt, ddln_kind, ddln_origin,
      ddln_sec, ddln_nsec, istat)
integer schdle, cpu, slot, ddln_halt, ddln_kind, ddln_origin, ddln_sec, ddln_nsec, istat
character* (*) name
```

パラメータ

schdle **fbsconfigure(3f)**または**fbsid(3f)**ライブラリ・ルーチンの呼び出しから取得するもしくは-1を設定。-1はFBSでスケジュールされたプロセスがスケジューラーの識別子を認識せずにそれがスケジュールされたFrequency-Based Schedulerを参照出来るようにします。

name プロセス(またはマルチスレッド化したプロセスのスレッド)を特定するパス名称。名称が全て空白である場合、*slot*領域(Frequency-Based SchedulerプロセスID)が提供される必要があります。

cpu ビット・マスクもしくは0または-1を設定のいずれか。ビット・マスクを指定した場合、(*cpu* & (1 << *i*))が設定されたプロセッサーが要求されます。*cpu*が0場合、呼び出しが行われるプロセッサーが要求されます。*cpu*が-1場合、全てのプロセッサーが要求されます。要求されたプロセッサーの1つで現在実行中の*name*で指定された最初のプロセスが返されます。

slot プロセスのFrequency-Based SchedulerプロセスID。スロット番号が-1である場合、名称とプロセッサーIDが提供される必要があります。

ddln_halt デッドライン違反の検出でスケジューラーは停止するかどうかを示します。この値は次のいずれかが可能です：

0 DL_NOHALT
デッドライン違反の検出でスケジューラーは停止すべきではないことを示します。

1 DL_HALT
デッドライン違反の検出でスケジューラーは停止すべきであることを示します。

ddln_kind プロセスに設定されるデッドラインの種類。この値は*ddln_sec*と*ddln_nsec*の値の解釈に影響します。*ddln_kind*は次のいずれかが可能です：

0 DEADLINE_CLEAR
指定したプロセスに現在割り当てられているデッドラインが削除されることを示します。指定したプロセスはもはやデッドラインに依存しません。これがデフォルトです。

1 DEADLINE_WALL_TIME
*ddln_sec*と*ddln_nsec*の値を同時にプロセスまたはスレッドに割り当てられたCLOCK_MONOTOICデッドライン時間に指定することを示します。

ddln_origin デッドライン時間が測定されるポイントを示します。これは次のいずれかが可能です：

0 DL_CYCLE_RELATIVE
スケジュールされたタスクのサイクルの開始からデッドライン時間が測定されることを示します。測定された時間はプロセスが実行を開始する前のあらゆる遅延(通常のプロセス・ディスパッチ・レイテンシー、割り込みを処理するのに費やす時間、より高い優先度のタスクがCPUを解放するのを待つのに費やす時間)を含むことになります。

1 DL_TASK_RELATIVE
スケジュールされたタスクが**fbswait(3f)**を終了して実行を開始する時間からデッドライン時間が測定されることを示します。

ddln_sec
ddln_nsec 負の値ではなく、デッドライン時間を表す秒とナノ秒の数字(C言語のtimespec構造体の値のように)を一緒に定義する必要があります。デッドライン時間は、指定したプロセスが**fbswait(3f)**に戻ることを要求するまでの指定した起点に対する最大時間です。デフォルトでこの値はゼロ秒、ゼロ・ナノ秒です。すなわち、プロセスがデッドラインを決して設定しない場合、これはゼロとなります。

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するにはmanページを参照して下さい。

sched_pgm_set_soft_overrun_limit - ソフト・オーバーラン上限を設定

本サブルーチンは Frequency-Based Scheduler に現在スケジュールされているプロセスの連続的なソフト・オーバーラン上限を設定します。

連続的なソフト・オーバーラン上限を設定するには、呼び出し元プロセスはスケジューラーに対する変更許可を持っている必要があります。

プロセスは以下の方法のいずれかで識別することが可能です：

- スロットのみ(*name*が空白の場合)
- パス名称とプロセッサーIDのペアのみ(*slot*が-1の場合)
- スロットおよびパス名称とプロセッサーIDのペアの両方

呼出し命令

```
call sched_pgm_set_soft_overrun_limit (schdle, name, cpu, slot, soft_limit, istat)
integer schdle, cpu, slot, soft_limit, istat
character* (*) name
```

パラメータ

schdle — **fbsconfigure(3f)**または**fbsid(3f)**ライブラリ・ルーチンの呼び出しから取得するもしくは-1を設定。-1はFBSでスケジュールされたプロセスがスケジューラーの識別子を認識せずにそれがスケジュールされたFrequency-Based Schedulerを参照出来るようにします。

name — プロセスを特定するパス名称。名称が全て空白である場合、*slot*領域(Frequency-Based SchedulerプロセスID)が提供される必要があります。

cpu — ビット・マスクもしくは0または-1を設定のいずれか。ビット・マスクを指定した場合、(*cpu* & (1 << *i*))が設定されたプロセッサーが要求されます。*cpu*が0場合、呼び出しが行われるプロセッサーが要求されます。*cpu*が-1場合、全てのプロセッサーが要求されます。要求されたプロセッサーの1つで現在実行中の*name*で指定された最初のプロセスにそのソフト・オーバーラン上限が設定されます。

slot — プロセスのFrequency-Based SchedulerプロセスID。スロット番号が-1である場合、名称とプロセッサーIDが提供される必要があります。

soft_limit — 失敗前に連続的なソフト・オーバーランの発生が許可された数。*soft_limit*は負ではない値でINT_MAX未満である必要があります。デフォルトでこの値はゼロです。プロセスが連続的なソフト・オーバーラン上限を決して設定しない場合、これはゼロとなります。

istat — エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するにはmanページを参照して下さい。

sched_pgm_soft_overrun_query - ソフト・オーバーラン処理を問合せ

本サブルーチンは Frequency-Based Scheduler に現在スケジュールされているプロセスまたはスレッドに関するソフト・オーバーラン処理の状態を問い合わせます。プロセスは以下の方法のいずれかで識別することが可能です：

- スロットのみ(*name*が空白の場合)
- パス名称とプロセッサーIDのペアのみ(*slot*が-1の場合)
- スロットおよびパス名称とプロセッサーIDのペアの両方

呼出し命令

```
call sched_pgm_soft_overrun_query （schdle, name, cpu, slot, soft_limit, soft_total,
     istat）
integer schdle, pu, slot, soft_limit, soft_total, istat
character* (*) name
```

パラメータ

schdle — **fbsconfigure(3f)**または**fbsid(3f)**ライブラリ・ルーチンの呼び出しから取得するもしくは-1を設定。-1はFBSでスケジュールされたプロセスがスケジューラーの識別子を認識せずにそれがスケジュールされたFrequency-Based Schedulerを参照出来るようにします。

name — プロセスを特定するパス名称。名称が全て空白である場合、*slot*領域(Frequency-Based SchedulerプロセスID)が提供される必要があります。

cpu — ビット・マスクもしくは0または-1を設定のいずれか。ビット・マスクを指定した場合、(*cpu* & (1 << *i*))が設定されたプロセッサーが要求されます。*cpu*が0場合、呼び出しが行われるプロセッサーが要求されます。*cpu*が-1場合、全てのプロセッサーが要求されます。要求されたプロセッサーの1つで現在実行中の*name*で指定された最初のプロセスが返されます。

slot — プロセスのFrequency-Based SchedulerプロセスID。スロット番号が-1である場合、名称とプロセッサーIDが提供される必要があります。

soft_limit — **sched_pgm_set_soft_overrun_limit(3f)**にを呼び出すことで設定される連続的なソフト・オーバーランの数。

soft_total — プロセスにより被ったソフト・オーバーランの総数。

istat — エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するにはmanページを参照して下さい。

schedfbsqry - Frequency-Based Schedulerのプロセスを問合せ

本サブルーチンは Frequency-Based Scheduler にスケジュールされたプロセスに関する情報を取得するために呼び出されます。情報はユーザーが指定したプロセッサーにスケジュールされた全てのプロセスが返されます。各プロセスに関して提供される情報は以下を含みます：

- プロセスを実行することが可能なCPU
- Frequency-Based SchedulerプロセスID
- スケジューリング優先度
- 周期(プロセスが各メジャー・フレームで起動される頻度を示すマイナー・サイクルの数)
- 開始ベース・サイクル(プロセスが各メジャー・フレームで起動される予定の最初のマイナー・サイクル)
- 「halt on overrun」フラグの値

呼出し命令

```
call schedfbsqry (schdle, cpu, buf1size, buf1, maxsize, buf2size, buf2, istat)
integer schdle, cpu, buf1size, buf1(buf1size), maxsize, buf2size, istat
character* (*) buf2
```

パラメータ

schdle スケジューリング情報を取得したいFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

cpu スケジューリング情報を取得するプロセッサーを示す整数値。許容値および対応する結果は次のとおりとなります：

- 0 呼び出しが行われたプロセッサー上で実行中のプロセスに関するスケジューリング情報が返されます
- -1 スケジューラー上の全てのプロセスに関するスケジューリング情報が返されます
- Bit mask (*cpu* & (1 << *i*))が設定された場合(*i*はゼロから15の範囲の整数でCPUを表します)、CPU *i*で実行中のプロセスに関するスケジューリング情報が返されます

buf1size *buf1*で表される配列のサイズを32bitワード単位で示す整数値。9ワードの情報がプロセスごとに返されるので、この値は9の倍数とすることを推奨します。

buf1 *cpu*パラメータで指定したプロセッサー上の各プロセスの連続する9個の整数値を含む配列。これらの値が返されるプロセスの数は*buf1size*パラメータの値に制約されます。例えば、*buf1size*の値が145である場合、16プロセスの値が返されます。これらの値はプロセスのスケジューリング情報を表します。単一プロセスにおける各配列エレメントに返される情報の種類を表7-6に示します。

表7-6 配列エレメントの中身: scedfbsqry

エレメント	中身
1	*buf2*内のプロセスのパス名称のバイト・オフセット
2	プロセスのパス名称のバイト長
3	プロセスが実行可能なCPUのマスク
4	プロセスのFrequency-Based SchedulerプロセスID
5	プロセスのスケジューリング・ポリシー
6	プロセスのスケジューリング優先度
7	プロセスが各メジャー・フレームで起動される頻度(期間)を示すマイナー・サイクルの数
8	プロセスが各メジャー・フレームで起動される予定の最初のマイナー・サイクル(開始ベース・サイクル)
9	「halt on overrun」フラグの値。ゼロ以外の値はフラグが設定されたことを示します。値ゼロはフラグが設定されていないことを示します。

maxsize *buf2*に返されるパス名称の最大長を示す整数値

buf2size *buf2*で表される文字列のサイズをバイト単位で示す整数値。*buf2*が取得したい全てのプロセスの名称を収容するのに十分な大きさであることを確実にするには、スケジューラーで許可されたプロセスの最大数(7-9ページに示された**fbsconfigure**の情報を参照して下さい)に32を掛けることで必要となるバイト数を計算するのに有用であると気付くかもしれません。

buf2 **schedfbsqry**が*cpu*パラメータで指定したプロセッサー上の各プロセスのパス名称を返す変数を参照します。パス名称は一続きの文字列として返されます。各文字列の長さは*maxsize*の値以下です。フル・パス名称を収容するのに十分な大きさではない場合、途中で終わったコンポーネント名称が返されます。返されるパス名称の数は*buf2size*パラメータの値に制約されます。

istat	**schedfbsqry**がエラーが発生したかどうかを示す整数値を返す変数を参照します。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**schedfbsqry(3f)**のmanページを参照して下さい。

schedpgmadd - Frequency-Based Schedulerにプロセスをスケジュール

schedpgmadd サブルーチンは新しいプロセスを生成して Frequency-Based Scheduler にそれをスケジュールするために呼び出されます。引数を含めるには、7-57 ページの **schedpgmadd_args** を参照して下さい。本サブルーチンを使用するには、呼び出し元プロセスは *fbscheduser* ロール(第 1 章を参照)に関連付けられた機能を所有している必要があることに注意することが重要です。

呼出し命令

```
call schedpgmadd (schdle, name, cid, prior, param, period, cycle, ab, cpu, slot, istat)
integer schdle, cid, prior, param, period, cycle, ab, cpu, slot, istat
character* (*) name
```

パラメータ

schdle — Frequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

name — スケジューラーにスケジュールするプログラムを特定する標準パス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。

cid — 指定したプロセスがスケジュールされるPOSIXスケジューリング・ポリシー(SCHED_FIFO, SCHED_RR, SCHED_OTHER)を示す整数値。スケジューリング・ポリシーはファイル**<sched.h>**に定義されています。

prior — 指定したプログラムのスケジューリング優先度を示す整数値。許容可能な優先度の値の範囲は指定するスケジューリング・ポリシーに影響されます。

各ポリシー(SCHED_FIFO, SCHED_RR, SCHED_OTHER)に関連付けられている優先度の許容範囲は、シェルからオプションや引数を指定することなく**run(1)**コマンドを起動(本コマンドの説明については対応するmanページを参照して下さい)することで判断することが可能です。より高い数値はより有益なスケジューリング優先度に対応します。

スケジューリング・ポリシーと優先度に関する詳細については *RedHawk Linux User's Guide*を参照して下さい。

param — Frequency-Based Schedulerにスケジュールされるプロセスに渡される整数値。この値は**rtparm**への呼び出し(本サブルーチンの説明は7-43ページを参照)を通してFBSでスケジュールされたプロセスで回収することが可能です。

period — 指定したプログラムが各メジャー・フレームで起動される頻度を示す整数値。頻度1は指定したプログラムがマイナー・サイクル毎に起動されることを示し、頻度2はマイナー・サイクル2回毎に1回、頻度3はマイナー・サイクル3回毎に1回起動されることを示します。

この値は1から**fbsconfigure** (7-9ページを参照)の呼び出しで定義した時に指定したスケジューラーのフレームを構成するマイナー・サイクルの数の範囲とすることが可能です。

cycle 指定したプロセスが各フレームで起動するようにスケジュールされている最初のマイナー・サイクルを示す整数値。この値はゼロからフレームあたりのマイナー・サイクルの総数-1の範囲とすることが可能です。フレームあたりのマイナー・サイクルの総数は、**fbsconfigure** (本サブルーチンの説明は7-9ページを参照)の呼び出しで指定されています。

ab 指定したプロセスでフレーム・オーバーランが発生した場合にスケジューラーを停止させるべきかどうかを示す整数値を含んでいるフラグ。ゼロ以外の値はスケジューラーを停止することを示します。

cpu 指定したプログラムの実行をスケジュール可能なプロセッサーを識別するマスク。許容値および対応する結果は次のとおり：

0	*name*で指定されたプログラムは呼び出しが行われるプロセッサーにスケジュールすることが可能
-1	*name*で指定されたプログラムはいずれのプロセッサーでもスケジュールすることが可能
Bit mask	(*cpu* & (1 << *i*))が設定された場合(*i*はゼロから15の範囲の整数でCPUを表します)、*name*で指定されたプログラムはCPU *i*にスケジュールすることが可能

slot スケジュールされたプロセスの一意のFrequency-Based SchedulerプロセスIDである整数値。

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**schedpgmadd(3f)**のmanページを参照して下さい。

schedpgmadd_args - Frequency-Based Schedulerに引数を付けてプロセスをスケジュール

schedpgmadd_args サブルーチンは引数と共に新しいプロセスを生成して Frequency-Based Scheduler にそれをスケジュールするために呼び出されます。本サブルーチンを使用するには、呼び出し元プロセスは *fbscheduser* ロール(第 1 章を参照)に関連付けられた機能を所有している必要があることに注意することが重要です。

呼出し命令

```
call schedpgmadd_args (schdle, name, arglist, arglen, argc, cid, prior, param, period, cycle,
ab, cpu, slot, istat)
integer schdle, cid, prior, param, period, cycle, ab, cpu, slot, istat
integer argc, arglen(0:argc-1)
character* (*) arglist(0:argc-1)
character* (*) name
```

パラメータ

schdle — Frequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

name — スケジューラーにスケジュールするプログラムを特定する標準パス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。

cid — 指定したプロセスがスケジュールされるPOSIXスケジューリング・ポリシー(SCHED_FIFO, SCHED_RR, SCHED_OTHER)を示す整数値。スケジューリング・ポリシーはファイル**<sched.h>**に定義されています。

prior — 指定したプログラムのスケジューリング優先度を示す整数値。許容可能な優先度の値の範囲は指定するスケジューリング・ポリシーに影響されます。

各ポリシー(SCHED_FIFO, SCHED_RR, SCHED_OTHER)に関連付けられている優先度の許容範囲は、シェルからオプションや引数を指定することなく**run(1)**コマンドを起動(本コマンドの説明については対応するmanページを参照して下さい)することで判断することが可能です。より高い数値はより有益なスケジューリング優先度に対応します。

スケジューリング・ポリシーと優先度に関する詳細については *RedHawk Linux User's Guide*を参照して下さい。

param — Frequency-Based Schedulerにスケジュールされるプロセスに渡される整数値。この値は**rtparm** (本サブルーチンの説明は7-43ページを参照)への呼び出しを通してFBSでスケジュールされたプロセスで回収することが可能です。

period 指定したプログラムが各メジャー・フレームで起動される頻度を示す整数値。頻度1は指定したプログラムがマイナー・サイクル毎に起動されることを示し、頻度2はマイナー・サイクル2回毎に1回、頻度3はマイナー・サイクル3回毎に1回起動されることを示します。この値は1から**fbsconfigure** (7-9ページを参照)の呼び出しで定義した時に指定したスケジューラーのフレームを構成するマイナー・サイクルの数の範囲とすることが可能です。

cycle 指定したプロセスが各フレームで起動するようにスケジュールされている最初のマイナー・サイクルを示す整数値。この値はゼロからフレームあたりのマイナー・サイクルの総数-1の範囲とすることが可能です。フレームあたりのマイナー・サイクルの総数は、**fbsconfigure** (本サブルーチンの説明は7-9ページを参照)の呼び出しで指定されています。

ab 指定したプロセスでフレーム・オーバーランが発生した場合にスケジューラーを停止させるべきかどうかを示す整数値を含んでいるフラグ。ゼロ以外の値はスケジューラーを停止することを示します。

cpu 指定したプログラムの実行をスケジュール可能なプロセッサーを識別するマスク。許容値および対応する結果は次のとおり：

0	*name*で指定されたプログラムは呼び出しが行われるプロセッサーにスケジュールすることが可能
-1	*name*で指定されたプログラムはいずれのプロセッサーでもスケジュールすることが可能
Bit mask	(*cpu* & (1 << *i*))が設定された場合(*i*はゼロから15の範囲の整数でCPUを表します)、*name*で指定されたプログラムはCPU *i*にスケジュールすることが可能

arglist, arglen, argc *arglist*配列の長さは*argc*に明確に渡されます。*arglist*と*arglen*は少なくとも*argc*と同じ次元数である必要があります。これらの配列は要求よりも大きな次元数である可能性がある一方、*argc*はルーチンで使用される各配列の先頭で要素の数を指定します。*arglist*の各要素の文字列長は*arglen*配列の対応する要素に渡されます。

slot スケジュールされたプロセスの一意のFrequency-Based SchedulerプロセスIDである整数値。

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**schedpgmadd_args(3f)**のmanページを参照して下さい。

schedpgmqry - Frequency-Based Schedulerのプロセスを問合せ

schedpgmqry サブルーチンは Frequency-Based Scheduler にスケジュールされた特定のプロセスに関する情報を取得するために呼び出されます。以下の方法のいずれかを使ってプロセスを特定することが可能です：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based SchedulerプロセスID(スロット番号)を指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based Scheduler プロセスIDを指定

NOTE

同じ CPU 上に複数回スケジュールされたプロセスを特定することが可能な唯一の方法は、その Frequency-Based Scheduler プロセス ID を指定することです。

返される情報は以下を含みます：

- プロセスのパス名称
- プロセスを実行することが可能なCPU
- Frequency-Based SchedulerプロセスID
- スケジューリング・ポリシー
- スケジューリング優先度
- 周期(プロセスが各メジャー・フレームで起動される頻度を示すマイナー・サイクルの数)
- 開始ベース・サイクル(プロセスが各メジャー・フレームで起動される予定の最初のマイナー・サイクル)
- 「halt on overrun」フラグの値

呼出し命令

```
call schedpgmqry (schdle, name, cpu, slot, cid, prior, period, cycle, ab, istat)
integer schdle, cpu, slot, cid, prior, period, cycle, ab, istat
character* (*) name
```

パラメータ

schdle スケジュールされたスケジューリング情報を取得したいプロセスのFrequency-Based Schedulerの識別子を表す一意の正の整数。**`fbsconfigure(3f)`** (本サブルーチンの説明は7-9ページを参照)または**`fbsid(3f)`** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

name 情報が返されるプロセスを特定する標準パス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。本変数が空白である場合、*slot*パラメータにFrequency-Based Schedulerプロセスを提供する必要があります。

cpu 情報が返されるプログラムを特定するために*name*パラメータの値と関連して使用されるプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
-1	いずれかのプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
Bit mask	(*cpu* & (1 << *i*))が設定され(*i*はゼロから15の範囲の整数でCPUを表します)、それが唯一のビット・セットである場合、CPU *i*で実行中の*name*で指定された最初のプロセスが指定されます
	(*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name*で指定された最初のプロセスが指定されます

slot 情報が返されるプロセスに関する一意のFrequency-Based Schedulerプロセスを提供する整数値。この値は**`schedpgmadd`** (本サブルーチンの説明は7-55ページを参照)の呼び出しを行ったときに取得します。*name*と*cpu*を指定するだけで問い合わせるプログラムを特定したい場合、この値は-1にする必要があります。

cid 指定したプロセスがスケジュールされたスケジューリング・ポリシーを示す整数値。

prior 指定したプログラムのスケジューリング優先度を示す整数値。

period 指定したプログラムが各メジャー・フレームで起動される頻度を示す整数値。

cycle 指定したプロセスが各フレームで起動するようにスケジュールされている最初のマイナー・サイクルを示す整数値。

ab 「halt on overrun」フラグの値を示す整数値。ゼロ以外の値はフラグが設定されたことを示します。値ゼロはフラグが設定されていないことを示します。

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**`schedpgmqry(3f)`**のmanページを参照して下さい。

schedpgmresched - プロセスを再スケジュール

schedpgmresched サブルーチンは Frequency-Based Scheduler にスケジュールされているプロセスに関するスケジューリング・パラメータを変更するために呼び出されます。例えば、プログラムのポリシーや優先度もしくは実行がスケジュールされている頻度を変更したくなる場合があります。しかしながら、スケジュールされた CPU を変更することは出来ません。

次のいずれかを望む場合、(1)スケジューリング・ポリシーを SCHED_FIFO または SCHED_RR ポリシーへ変更、(2)SCHED_FIFO または SCHED_RR ポリシーでスケジュールされたプロセスの優先度を変更、(3)プロセスごとの上限を超えて SCHED_OTHER ポリシーでスケジュールされたプロセスの優先度を上げる、以下の条件を満たす必要があります：

- 呼び出し元プロセスは*fbscheduser*ロールに関連付けられた機能を所有している必要があります(第1章を参照)
- 呼び出し元プロセスの実効ユーザーIDはターゲット・プロセス(スケジューリング・ポリシーと優先度が設定されているプロセス)の実効ユーザーIDと一致している必要があります。

以下のいずれかの方法を使って再スケジュールするプロセスを特定することが可能です：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based SchedulerプロセスID(スロット番号)を指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based SchedulerプロセスIDを指定

NOTE

同じ CPU 上で複数回スケジュールされているプロセスを識別するために使用することが可能な唯一の方法は、その Frequency-Based Scheduler プロセス ID を指定することです。

呼出し命令

```
call schedpgmresched (schdle, name, cpu, slot, cid, prior, param, period, cycle, ab, istat)
integer schdle, cpu, slot, cid, prior, param, period, cycle, ab, istat
character* (*) name
```

パラメータ

schdle プロセスがスケジュールされているFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

name 再スケジュールするプロセスを特定する標準パス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。

本変数が空白である場合、*slot*パラメータにFrequency-Based Schedulerプロセスを提供する必要があります。

cpu

再スケジュールするプロセスを特定するために*name*パラメータの値と関連して使用されるプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが再スケジュールされます
-1	いずれかのプロセッサーで現在実行中の*name*で指定された最初のプロセスが再スケジュールされます
Bit mask	(*cpu* & (1 << *i*))が設定され(*i*はゼロから15の範囲の整数でCPUを表します)、それが唯一のビット・セットである場合、CPU *i*で実行中の*name*で指定された最初のプロセスが再スケジュールされます
	(*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name*で指定された最初のプロセスが再スケジュールされます

slot

再スケジュールするプロセスに関する一意のFrequency-Based SchedulerプロセスIDを提供する整数値。この値は**schedpgmadd** (本サブルーチンの説明は7-55ページを参照)の呼び出しを行ったときに取得します。*name*と*cpu*を指定するだけで問い合わせるプログラムを特定したい場合、この値は-1にする必要があります。

cid

指定したプログラムがスケジュールされるスケジューリング・ポリシー(SCHED_FIFO, SCHED_RR, SCHED_OTHER)を示す整数値。スケジューリング・ポリシーはファイル**<sched.h>**に定義されています。

prior

指定したプログラムのスケジューリング優先度を示す整数値。許容可能な優先度の値の範囲は指定するスケジューリング・ポリシーに影響されます。

各ポリシー(SCHED_FIFO, SCHED_RR, SCHED_OTHER)に関連付けられている優先度の許容範囲は、シェルからオプションや引数を指定することなく**run(1)**コマンドを起動(本コマンドの説明については対応するmanページを参照して下さい)することで判断することが可能です。より高い数値はより有益なスケジューリング優先度に対応します。スケジューリング・ポリシーと優先度に関する詳細については*RedHawk Linux User's Guide*を参照して下さい。

param

Frequency-Based Schedulerにスケジュールされるプロセスに渡される整数値。

period

指定したプログラムが各メジャー・フレームで起動される頻度を示す整数値。頻度1は指定したプログラムがマイナー・サイクル毎に起動されることを示し、頻度2はマイナー・サイクル2回毎に1回、頻度3はマイナー・サイクル3回毎に1回起動されることを示します。

この値は1から**fbsconfigure** (7-9ページを参照)の呼び出しで定義した時に指定したスケジューラーのフレームを構成するマイナー・サイクルの数の範囲とすることが可能です。

cycle 指定したプロセスが各フレームで起動するようにスケジュールされている最初のマイナー・サイクルを示す整数値。この値はゼロからフレームあたりのマイナー・サイクルの総数-1の範囲とすることが可能です。フレームあたりのマイナー・サイクルの総数は、**fbsconfigure** (本サブルーチンの説明は7-9ページを参照)の呼び出しで指定されています。

ab 指定したプロセスでフレーム・オーバーランが発生した場合にスケジューラーを停止させるべきかどうかを示す整数値を含んでいるフラグ。ゼロ以外の値はスケジューラーを停止することを示します。

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**schedpgmresched(3f)**のmanページを参照して下さい。

Performance Monitorルーチン

Performance Monitor ルーチンは Performance Monitor の主要な機能へのアクセスを提供します。それらは以下のような基本的な操作を実行することを可能にします：

- プロセスまたはプロセッサーのPerformance Monitor値を消去
- プロセスまたはプロセッサーのPerformance Monitorを開始および停止
- プロセスまたはプロセッサーのPerformance Monitor値を取得

ルーチンの要約

Performance Monitor ルーチンが表 7-7 に要約されています。各ルーチンに関する完全な情報は「Performance Monitor ルーチンの使用」項で提供されます。

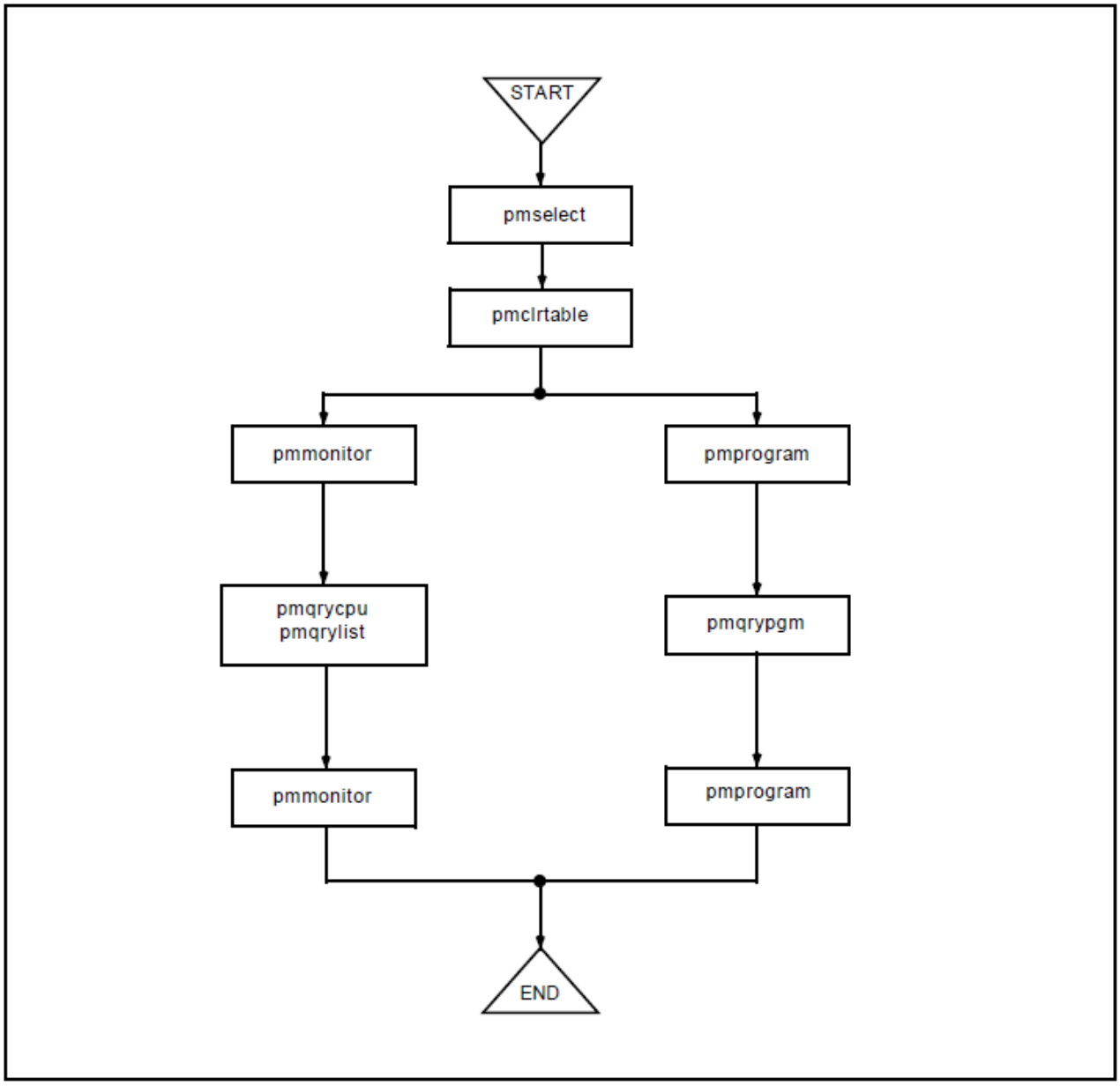
表7-7 Performance MonitorのFORTRANライブラリ・ルーチン

ルーチン	ページ	説明
pmclrpgm	7-67	プロセスの値を消去
pmclrtable	7-69	プロセッサーの値を消去
pmmonitor	7-70	プロセッサーのPerformance Monitorを開始/停止
pmprogram	7-71	プロセスのPerformance Monitorを開始/停止
pmqrycpu	7-73	選択したプロセッサーの値を問合せ
pmqrylist	7-75	プロセスのリストの値を問合せ
pmqrypgm	7-77	選択したプロセスの値を問合せ
pmqrytimer	7-80	Performance Monitorのモードを問合せ
pmselect	7-81	Performance Monitorのモードを選択

FORTRANライブラリ・コールの順序

アプリケーション・プログラムから Performance Monitor ルーチンを呼び出す可能性のあるおおよその順番を図 6-2 に示します。

図7-2 FORTRANライブラリ・コールの順序：Performance Monitor



Performance Monitorの使用

これ以降は、**libF77rt** ライブラリ内に含まれる Performance Monitor ルーチンをアルファベット順で示します。

pmclrpgm - プロセスの値を消去

本サブルーチンは Frequency-Based Scheduler にスケジュールされた特定のプロセスの Performance Monitor の値を消去するために呼び出されます。以下の方法のいずれかを使ってプロセスを特定することが可能です：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based SchedulerプロセスID(スロット番号)を指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based SchedulerプロセスIDを指定

NOTE

本ルーチンはプロセスのソフト・オーバーラン総数を消去します。

呼出し命令

```
call pmclrpgm (schdle, name, cpu, slot, istat)
integer schdle, cpu, slot, istat
character* (*) name
```

パラメータ

schdle プロセスがスケジュールされているFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

name 値を消去するプロセスを特定する標準パス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。本変数が空白である場合、*slot*パラメータにFrequency-Based SchedulerプロセスIDを提供する必要があります。

cpu 値を消去するプロセスを特定するために*name*パラメータの値と関連して使用されるプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
-1	いずれかのプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます

	Bit mask	(*cpu* & (1 << *i*))が設定され(*i*はゼロから15の範囲の整数でCPUを表します)、それが唯一のビット・セットである場合、CPU *i*で実行中の*name*で指定された最初のプロセスが指定されます (*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name*で指定された最初のプロセスが指定されます
slot		値を消去するプロセスに関する一意のFrequency-Based Schedulerプロセスを提供する整数値。この値は**schedpgmadd** (本サブルーチンの説明は7-55ページを参照)の呼び出しを行ったときに取得します。*name*と*cpu*を指定するだけで問い合わせるプログラムを特定したい場合、この値は-1にする必要があります。
istat		エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**pmclrpgm(3f)**のmanページを参照して下さい。

pmclrtable - プロセッサーの値を消去

本サブルーチンは選択したスケジューラーにおいて指定した1つ以上のプロセッサー上のFBSでスケジュールされたプロセスのPerformance Monitorの値を消去するために呼び出されます。

NOTE

本サブルーチンは全ての関連するプロセスのソフト・オーバーラン総数を消去します。

呼出し命令

```
call pmclrtable (schdle, cpucount, cpulist, istat)
integer schdle, cpucount, cpulist(cpucount), istat
```

パラメータ

schdle プロセスがスケジュールされているFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

cpucount *cpulist*で表される配列に含まれている要素の数を示す整数値。

cpulist *cpucount*パラメータで指定された要素数で構成され、Performance Monitorの値を消去するプロセッサーを示す1つ以上の整数値を含む配列。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサー上で実行中のFBSでスケジュールされたプロセスのPerformance Monitorの値が消去されます
-1	スケジューラー上の全てのプロセスのPerformance Monitorの値が消去されます
Bit mask	(*cpu* & (1 << *i*))が設定された場合(*i*はゼロから15の範囲の整数でCPUを表します)、CPU *i*で実行中のプロセスのPerformance Monitorの値が消去されます

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**pmclrtable(3f)**のmanページを参照して下さい。

pmmonitor - プロセッサーのPerformance Monitorを開始/停止

本サブルーチンは選択したスケジューラーにおいて指定した1つ以上のプロセッサー上のFBSでスケジュールされたプロセスの Performance Monitor を開始または停止するために呼び出されます。

呼出し命令

```
call pmmonitor （schdle, pmflag, cpucount, cpulist, istat）
integer schdle, pmflag, cpucount, cpulist(cpucount), istat
```

パラメータ

schdle プロセスがスケジュールされているFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

pmflag Performance Monitorを開始するのか停止するのかを示す整数値。ゼロ以外の値はPerformance Monitorを開始することを示します。値ゼロはPerformance Monitorを停止することを示します。

cpucount *cpulist*で表される配列の要素の数を示す整数値。

cpulist *cpucount*パラメータで指定された要素数で構成され、Performance Monitorの値を開始または停止するプロセッサーを示す1つ以上の整数値を含む配列。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサー上で実行中のFBSでスケジュールされたプロセスのPerformance Monitorが開始または停止されます
-1	スケジューラー上の全てのプロセスのPerformance Monitorが開始または停止されます
Bit mask	(*cpu* & (1 << *i*))が設定された場合(*i*はゼロから15の範囲の整数でCPUを表します)、CPU *i*で実行中のプロセスのPerformance Monitorが開始または停止されます

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**pmmonitor(3f)**のmanページを参照して下さい。

pmprogram - プロセスのPerformance Monitorを開始/開始

本サブルーチンは Frequency-Based Scheduler にスケジュールされた特定のプロセスの Performance Monitor を開始または停止するために呼び出されます。以下の方法のいずれかを使ってプロセスを特定することが可能です：

- プロセスの名称とそれがスケジュールされているCPUを指定
- プロセスのFrequency-Based SchedulerプロセスID(スロット番号)を指定
- プロセスの名称、それがスケジュールされているCPU、そのFrequency-Based Scheduler プロセスIDを指定

呼出し命令

```
call pmprogram (schdle, name, cpu, slot, pmflag, istat)
integer schdle, cpu, slot, pmflag, istat
character* (*) name
```

パラメータ

schdle プロセスがスケジュールされているFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

name Performance Monitorを開始または停止するプロセスを特定する標準パス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。本変数が空白である場合、*slot*パラメータにFrequency-Based SchedulerプロセスIDを提供する必要があります。

cpu Performance Monitorを開始または停止するプロセスを特定するために*name*パラメータの値と関連して使用されるプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
-1	いずれかのプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
Bit mask	(*cpu* & (1 << *i*))が設定され(*i*はゼロから15の範囲の整数でCPUを表します)、それが唯一のビット・セットである場合、CPU *i*で実行中の*name*で指定された最初のプロセスが指定されます
	(*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name*で指定された最初のプロセスが指定されます

slot	Performance Monitorを開始または停止するプロセスに関する一意のFrequency-Based SchedulerプロセスIDを提供する整数値。この値は**schedpgmadd**(本サブルーチンの説明は7-55ページを参照)の呼び出しを行ったときに取得します。*name*と*cpu*を指定するだけで問い合わせるプログラムを特定したい場合、この値は-1にする必要があります。
pmflag	Performance Monitorを開始するのか停止するのかを示す整数値。ゼロ以外の値はPerformance Monitorを開始することを示します。値ゼロはPerformance Monitorを停止することを示します。
istat	エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**pmprogram(3f)**のmanページを参照して下さい。

pmqrycpu - 選択したプロセッサーの値を問合せ

本サブルーチンは選択したスケジューラーにおいて1つ以上の指定したプロセッサー上のFBSでスケジュールされたプロセスのPerformance Monitorの値を取得するために呼び出されます。

呼出し命令

```
call pmqrycpu (schdle, cpu, bufsiz, buf, istat)
integer schdle, cpu, bufsiz, buf(bufsiz), istat
```

パラメータ

schdle プロセスがスケジュールされているFrequency-Based Schedulerの識別子を表す一意の正の整数。**`fbsconfigure(3f)`** (本サブルーチンの説明は7-9ページを参照)または**`fbsid(3f)`** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

cpu Performance Monitorの値を取得するプロセッサーを示す整数値。許容値および対応する結果は次のとおりとなります：

- 0 呼び出しが行われたプロセッサー上で実行中のFBSでスケジュールされたプロセスに関するPerformance Monitorの値が返されます
- -1 スケジューラー上の全てのプロセスに関するPerformance Monitorの値が返されます
- Bit mask (*cpu* & (1 << *i*))が設定された場合(*i*はゼロから15の範囲の整数でCPUを表します)、CPU *i*で実行中のプロセスに関するPerformance Monitorの値が返されます

bufsiz *buf*で表される配列のサイズを32bitワード単位で示す整数値。16ワードの情報がプロセスごとに返されるので、この値は16の倍数とすることを推奨します。

buf *cpu*パラメータで指定したプロセッサー上のFBSでスケジュールされた各プロセスの連続する16個の整数値を含む配列。これらの値が返されるプロセスの数は*bufsiz*パラメータの値に制約されます。例えば、*bufsiz*の値が165である場合、10プロセスの値が返されます。これらの値はプロセスのPerformance Monitor情報を表します。単一プロセスにおける各配列エレメントに返される情報の種類を表7-8に示します。

表7-8 配列エレメントの中身: pmqrycpu

エレメント	中身
1	プロセスのFrequency-Based SchedulerプロセスID(スロット番号)
2	プロセスが最後にスケジューラーに起動されてから**fbswait**が呼ばれるまでに実行に費やした時間(最新時間)
3	プロセスがスケジューラーに起動された回数(合計反復数、またはサイクル数)
4	プロセスが全てのサイクルで実行に費やした秒数(秒単位の合計時間)。プロセスが実行に費やした時間の合計はエレメント4の値にエレメント5の値を加えたものに等しくなります。
5	プロセスが全てのサイクルで実行に費やした追加のマイクロ秒数(マイクロ秒単位の合計時間)。プロセスが実行に費やした時間の合計はエレメント4の値にエレメント5の値を加えたものに等しくなります。
6	プロセスに起因するハード・フレーム・オーバーランの数
7	プロセスが1サイクルで実行に費やした時間の最小(最小サイクル時間)
8	最小サイクル時間が発生したマイナー・サイクルの番号(最小サイクル・サイクル)
9	最小サイクル時間が発生したメジャー・フレームの番号(最小サイクル・フレーム)
10	プロセスが1サイクルで実行に費やした時間の最大(最大サイクル時間)
11	最大サイクル時間が発生したマイナー・サイクルの番号(最大サイクル・サイクル)
12	最大サイクル時間が発生したメジャー・フレームの番号(最大サイクル・フレーム)
13	プロセスが1メジャー・フレームの間で実行に費やした時間の最小(最小フレーム時間)
14	最小フレーム時間が発生したメジャー・フレームの番号(最小フレーム・フレーム)
15	プロセスが1メジャー・フレームの間で実行に費やした時間の最大(最大フレーム時間)
16	最大フレーム時間が発生したメジャー・フレームの番号(最大フレーム・フレーム)

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**pmqrycpu(3f)**のmanページを参照して下さい。

pmqrylist - プロセスのリストの値を問合せ

本サブルーチンは Frequency-Based Scheduler 上にスケジュールされたプロセスのリストの Performance Monitor の値を取得するために呼び出されます。

呼出し命令

```
call pmqrylist (schdle, slotcount, slotlist, bufsiz, buf, istat)
integer schdle, slotcount, slotlist(slotcount), bufsiz, buf(bufsiz), istat
```

パラメータ

schdle Performance Monitorの値が要求されるFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

slotcount *slotlist*で表される配列内に含まれるFrequency-Based SchedulerプロセスIDの数を示す整数値。

slotlist *slotcount*パラメータで指定された要素数で構成され、Performance Monitorの値が返されるFrequency-Based SchedulerプロセスIDを示す1つ以上の整数値を含む配列。

bufsiz *buf*で表される配列のサイズを32bitワード単位で示す整数値。15ワードの情報がプロセスごとに返されるので、この値は15の倍数とすることを推奨します。

buf FBSでスケジュールされた各プロセスの連続する15個の整数値を含む配列。これらの値が返されるプロセスの数は*bufsiz*パラメータの値に制約されます。例えば、*bufsiz*の値が155である場合、10プロセスの値が返されます。これらの値はプロセスのPerformance Monitor情報を表します。単一プロセスにおける各配列エレメントに返される情報の種類を表7-9に示します。

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**pmqrylist(3f)**のmanページを参照して下さい。

表7-9 配列エレメントの中身: pmqrylist

エレメント	中身
1	プロセスが最後にスケジューラーに起動されてから**fbswait**が呼ばれるまでに実行に費やした時間(最新時間)
2	プロセスがスケジューラーに起動された回数(合計反復数、またはサイクル数)
3	プロセスが全てのサイクルで実行に費やした秒数(秒単位の合計時間)。プロセスが実行に費やした時間の合計はエレメント3の値にエレメント4の値を加えたものに等しくなります。
4	プロセスが全てのサイクルで実行に費やした追加のマイクロ秒数(マイクロ秒単位の合計時間)。プロセスが実行に費やした時間の合計はエレメント3の値にエレメント4の値を加えたものに等しくなります。
5	プロセスに起因するハード・フレーム・オーバーランの数
6	プロセスが1サイクルで実行に費やした時間の最小(最小サイクル時間)
7	最小サイクル時間が発生したマイナー・サイクルの番号(最小サイクル・サイクル)
8	最小サイクル時間が発生したメジャー・フレームの番号(最小サイクル・フレーム)
9	プロセスが1サイクルで実行に費やした時間の最大(最大サイクル時間)
10	最大サイクル時間が発生したマイナー・サイクルの番号(最大サイクル・サイクル)
11	最大サイクル時間が発生したメジャー・フレームの番号(最大サイクル・フレーム)
12	プロセスが1メジャー・フレームの間で実行に費やした時間の最小(最小フレーム時間)
13	最小フレーム時間が発生したメジャー・フレームの番号(最小フレーム・フレーム)
14	プロセスが1メジャー・フレームの間で実行に費やした時間の最大(最大フレーム時間)
15	最大フレーム時間が発生したメジャー・フレームの番号(最大フレーム・フレーム)

pmqrypgm - 選択したプロセスの値を問合せ

本サブルーチンは Frequency-Based Scheduler にスケジュールされた特定のプロセスの Performance Monitor の値を取得するために呼び出されます。

呼出し命令

```
call pmqrypgm （schdle, name, cpu, slot, last, iter, totsec, totusec, over, minc, minctc, minctf,
      maxc, maxctc, maxctf, minf, minftf, maxf, maxftf, istat）
integer  schdle, cpu, slot, last, iter, totsec, totusec, over, minc, minctc, minctf, maxc, maxctc,
      maxctf, minf, minftf, maxf, maxftf, istat
character* （*） name
```

パラメータ

schdle Performance Monitorの値が要求されるFrequency-Based Schedulerの識別子を表す一意の正の整数。**fbsconfigure(3f)** (本サブルーチンの説明は7-9ページを参照)または**fbsid(3f)** (7-14ページを参照)の呼び出しを行うことでこの値を取得することが可能です。識別子を認識せずに呼び出すプロセスがスケジュールされているFrequency-Based Schedulerを参照するには、値-1を指定することが可能です。

name Performance Monitorの値が返されるプロセスを特定する標準パス名称。最大1024文字のフル・パスまたは相対パスの名称を指定することが可能です。本変数が空白である場合、*slot*パラメータにFrequency-Based SchedulerプロセスIDを提供する必要があります。

cpu Performance Monitorの値が返されるプロセスを特定するために*name*パラメータの値と関連して使用されるプロセッサを示す整数値。許容値および対応する結果は次のとおり：

0	呼び出しが行われたプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
-1	いずれかのプロセッサーで現在実行中の*name*で指定された最初のプロセスが指定されます
Bit mask	(*cpu* & (1 << *i*))が設定され(*i*はゼロから15の範囲の整数でCPUを表します)、それが唯一のビット・セットである場合、CPU *i*で実行中の*name*で指定された最初のプロセスが指定されます
	(*cpu* & (1 << *i*))が設定されそれが唯一のビット・セットではない場合、選択したCPUのいずれかで現在実行中の*name*で指定された最初のプロセスが指定されます

slot Performance Monitorの値が返されるプロセスに関する一意のFrequency-Based SchedulerプロセスIDを提供する整数値。この値は**schedpgmadd** (本サブルーチンの説明は7-55ページを参照)の呼び出しを行ったときに取得します。

	*name*と*cpu*を指定するだけで問い合わせるプログラムを特定したい場合、この値は-1にする必要があります。
last	プロセスが最後にスケジューラーに起動されてから**fbswait**が呼ばれるまでに実行に費やした時間を示す整数値(最新時間)。
iter	最後にPerformance Monitorの値が消去されPerformance Monitorが有効になってからプロセスがFrequency-Based Schedulerに起動された回数を示す整数値(合計反復数またはサイクル数)。
totsec	プロセスが全てのサイクルで実行に費やした秒数を示す整数値(秒単位の合計時間)。プロセスが実行に費やした時間の合計は*totsec*の値に*totusec*の値を加えたものに等しくなります。
totusec	プロセスが全てのサイクルで実行に費やした追加のマイクロ秒数を示す整数値(マイクロ秒単位の合計時間)。プロセスが実行に費やした時間の合計は*totsec*の値に*totusec*の値を加えたものに等しくなります。
over	プロセスがハード・フレーム・オーバーランを引き起こした回数を示す整数値。
minc	プロセスが1サイクルで実行に費やした時間の最小を示す整数値(最小サイクル時間)。
minctc	最小サイクル時間が発生したマイナー・サイクルの番号を示す整数値(最小サイクル・サイクル)。
minctf	最小サイクル時間が発生したメジャー・フレームの番号を示す整数値(最小サイクル・フレーム)。
maxc	プロセスが1サイクルで実行に費やした時間の最大を示す整数値(最大サイクル時間)。
maxctc	最大サイクル時間が発生したマイナー・サイクルの番号を示す整数値(最大サイクル・サイクル)。
maxctf	最大サイクル時間が発生したメジャー・フレームの番号を示す整数値(最大サイクル・フレーム)。
minf	プロセスが1メジャー・フレームの間で実行に費やした時間の最小を示す整数値(最小フレーム時間)。
minftf	最小フレーム時間が発生したメジャー・フレームの番号を示す整数値(最小フレーム・フレーム)。

maxf プロセスが1メジャー・フレームの間で実行に費やした時間の最大を示す整数値(最大フレーム時間)。

maxftf 最大フレーム時間が発生したメジャー・フレームの番号を示す整数値(最大フレーム・フレーム)。

istat エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**pmqrypgm(3f)**のmanページを参照して下さい。

pmqrytimer - Performance Monitorのモードを問合せ

本サブルーチンは Performance Monitor のタイミング値が割り込みの処理に費やす時間を含めるのか除外するのかを判定するために呼び出されます。

呼出し命令

```
call pmquerytimer (mode)
integer mode
```

パラメータ

mode Performance Monitorのタイミング値が割り込みの処理に費やす時間を含めるのか除外するのかを示す値。値1は割り込み時間を含まれていることを示します。値0は割り込み時間が除外されていることを示します。呼び出しが成功した場合は値1または0が返され、特定の種類のエラーが発生したことを示すために負の値が返されます。返される可能性のある値およびそれらが表すエラーの種類を表示するには **pmquerytimer(3f)** のmanページを参照して下さい。

pmselect - Performance Monitorのモードを選択

本サブルーチンは Performance Monitor を実行するタイミング・モードを選択するために呼び出されます。タイミング・モードは割り込みを処理するのに費やした時間を含めるもしくは除外することを設定することが可能です。

割り込み時間を含んでいる場合、プロセスのユーザーおよびシステム時間は、プロセスが現在処理を実行している時に割り込みを処理するのに費やした全ての時間を含めて生じた経過時間を合計します。割り込みを処理するのに費やした時間はプロセスのシステム時間に加算されます。

割り込み時間を除外している場合、プロセスのユーザーおよびシステム時間は、プロセスが現在処理を実行している時に割り込みを処理するのに費やした全ての時間を除外して生じた時間を合計します。

タイミング・モードを設定するには、呼び出し元プロセスは *fbscheduser* ロール(1 章を参照して下さい)に関連付けられたアクセス権を持っている必要があることに注意して下さい。

呼出し命令

```
call pmselect (mode, istat)
integer mode, istat
```

パラメータ

mode	割り込みを処理するのに費やした時間をPerformance Monitorのタイミング値に含めるのか除外するのかを示す整数値。ゼロ以外の値は割り込み時間が含まれることを示します。値ゼロは割り込み時間が除外されることを示します。
istat	エラーが発生したかどうかを示す整数値。値0はエラーが発生していないことを示します。ゼロ以外の値は特定の種類のエラーが発生したことを示します。返される可能性のあるゼロ以外の値およびそれらが表すエラーの種類を表示するには**`pmselect(3f)`**のmanページを参照して下さい。

付録A

rtcpスクリプトの実例

本付録には **rtcp** コマンドを実行するためにシステム・コマンド・プロンプトで呼び出すことが可能なスクリプトの実例が含まれています。

本スクリプトは以下を行うために使用するコマンドを例証します：

- スケジューラーを構成しプログラムをそこにスケジュール(**cs, sp**)
- FBSでスケジュールされたプロセスに関する情報を表示(**vp**)
- スケジューラーの構成を表示(**vs**)
- リアルタイム・クロックを接続、設定、開始(**ats, stc, rc**)
- Performance Monitorの値を消去(**cpm**)
- Performance MonitorとFrequency-Based Schedulerを開始(**pm, start**)
- マイナー・サイクルとメジャー・フレームの総数を表示(**vc**)
- Performance Monitorを停止し、リアルタイム・クロックとFrequency-Based Schedulerを停止(**pm, sc, stop**)
- リアルタイム・クロックを分離(**dts**)
- スケジューラーとスケジュールされた全てのプロセスを削除(**rms**)

```
rtcp cs -s37 -C10 -I664 -M5 -N10
rtcp sp -s37 -n ./program1 -c0 -bF -p19 -f4 -m0 -ohalt
rtcp sp -s37 -n ./program2 -c1 -bF -p19 -f2 -m1 -ohalt
rtcp vp -s37 -c*
rtcp vs -s37
rtcp ats -s37 -d/dev/rcim/rtc0
rtcp stc -s37 -O10000 -D1
rtcp rc -s37
rtcp cpm -s37 -c*
rtcp pm -s37 -c* -PON
rtcp start -s37
rtcp vc -s37
rtcp pm -s37 -c* -POFF
rtcp sc -s37
rtcp stop -s37
rtcp dts -s37
rtcp rms -s37 -a
```

付録B

rtcpエラー・メッセージ

本付録にはリアルタイム・コマンド・プロセッサー(**rtcp**)によって報告される可能性のあるエラーの説明が含まれています。

表B-1 rtcpエラー

エラー	説明
-2	割り込みデバイスが指定されていない
-3	EOCおよび割り込みデバイスの両方が指定されていない
-4	プロセスが指定されていない
-5	指定したrtcpコマンドが無効
-6	指定したヘルプ・コマンドが無効
-7	CPUパラメータ(-c)が無効
-8	頻度パラメータ(-f)が無効
-9	haltフラグ・パラメータ(-h | -o)が無効
-10	開始サイクル・パラメータ(-m)が無効
-11	優先度パラメータ(-p)が無効
-12	サイクル数パラメータ(-C)が無効
-13	クロック・ティック持続時間パラメータ(-D)が無効
-14	サイクルあたりのプロセス数パラメータ(-M)が無効
-15	Frequency-Based Schedulerあたりのプロセス数パラメータ(-N)が無効
-16	サイクルあたりのクロック・ティック数パラメータ(-O)が無効
-17	PMフラグ・パラメータ(-P)が無効
-18	指定したパラメータが無効
-22	rjファイルが指定されていない
-23	指定したrjファイルが無効
-24	指定したPMの表示モードが無効
-25	指定したPMのタイミング・モードが無効
-26	割り込み時間を除外するタイミング・モードに変更出来ない
-27	指定したスケジューリング・ポリシーが無効
-28	rtcpを終了
-29	ソフト・オーバーラン上限パラメータ(-L)が無効
-30	ホストの数(-H)が上限を超えた
-31	ホスト名称(-H)が指定されていない
-32	スケジューラーのリストが取得できない
-33	デッドライン・パラメータ(-T)が無効
-34	デッドライン起点パラメータ(-r)が無効 (サイクルまたはタスク)

付録C

Cインターフェースの実例

本付録には Frequency-Based Scheduler および Performance Monitor への C ライブラリ・インターフェースの使用を例証するプログラムの実例が含まれています。

プログラムは以下のタスクを実行します：

- スケジューラーを構成
- スケジューラーにプログラムをスケジュール
- リアルタイム・クロックを接続、設定、開始
- FBSでスケジュールされた各プロセスのPerformance Monitorを開始
- Frequency-Based Schedulerを開始
- FBSにスケジュールされたプロセスのPerformance Monitorの値を取得
- プロセッサーのアイドル時間を関し

プログラムの実例は次ページから始まります。

schedule.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sched.h>
#include <sys/stat.h>
#include <fbsched.h>
#include <errno.h>
#define NUM_PROCS 4
#define START 1
#define STOP 0
#define PROGRAM1 "./prog1"
#define PROGRAM2 "./prog2"
#define IDLE0 "idle0"
#define IDLE1 "idle1"
#define CLOCK1 "/dev/rcim/rtc0"
void
cleanup(int fbs_id)
{
    fbsrunrtc(fbs_id, 0); /* stop timing device */
    fbsdetach(fbs_id); /* detach timing device */
    fbsremove(fbs_id, -1); /* remove frequency-based scheduler */
}
int main()
{
    struct fbsconfig_ds fbs_buf;
    struct pgm2_ds sched_buf;
    struct pmqry_ds pm_buf[NUM_PROCS];
    struct fbsinfo_ds info_buf;
    struct fbscycle_ds cycle_buf;
    int idle0_fpid; /* fpid for idle on cpu 0 */
    int idle1_fpid; /* fpid for idle on cpu 1 */
    int pgm1_fpid; /* fpid for testprogram 1 */
    int pgm2_fpid; /* fpid for testprogram 2 */
    int cpu;
    int istat;
    int pmflg;
    int intrflg;
    int i;
    int count;
    int resolution;
    char name[1024]; /* programa??s full or orelative path name */
    char *pgmname[4]; /* programa??s name */
    FILE *fp;
    /* Open file to store performance information */
    fp= fopen("pmresults", "w+");
    if (fp == NULL)
       printf("open failed errno = %d¥n", errno);
    /*
    * CONFIGURE SCHEDULER
    */
    fbs_buf.key = 37; /* scheduler key */
    fbs_buf.cycles = 10; /* number of cycles per frame */
```

```
fbs_buf.progs = 5; /* max. number of programs per cycle */
fbs_buf.max = 10; /* max. number of programs allowed on the fbs */
fbs_buf.reset = -1; /* kill/remove processes currently scheduled */
/* owner/group read/write */
fbs_buf.configflg = 0664;
istat = fbsconfigure(&fbs_buf);
if (istat != 0) {
   printf("could not configure scheduler: errno = %d¥n" ,errno);
   exit(1);
}
/*
* SCHEDULE test program1 PROGRAM1
*/
sched_buf.name_ptr = PROGRAM1;
sched_buf.cid = SCHED_FIFO; /* first-in-first out (FIFO) policy */
sched_buf.prior = 19;
sched_buf.param = 0; /* optional initiation parameter */
sched_buf.period = 2; /* time between wakeups */
sched_buf.cycle = 0; /* starting base cycle */
sched_buf.halt = 0; /* halt on overrun */
/* Set cpu mask to schedule testprogram1 on cpu 0 */
sched_buf.cpu = 1;
istat = sched_pgmadd(fbs_buf.fbs_id, &sched_buf);
printf("fbsid = %d¥n", fbs_buf.fbs_id);
if (istat != 0) {
   printf("could not schedule %s on cpu %d : errno = %d¥n",
   sched_buf.name_ptr, sched_buf.cpu>>1, errno);
   cleanup(fbs_buf.fbs_id);
   return(1);
}
pgm1_fpid = sched_buf.fpid;
printf("pgm1 fpid = %d¥n", pgm1_fpid);
/*
* SCHEDULE test program2 PROGRAM2
*/
sched_buf.name_ptr = PROGRAM2;
sched_buf.prior = 19;
sched_buf.cid = SCHED_FIFO; /* first-in-first out (FIFO) policy */
sched_buf.param = 0; /* optional initiation parameter */
sched_buf.period = 2; /* time between wakeups */
sched_buf.cycle = 1; /* starting base cycle */
sched_buf.halt = 0; /* halt on overrun */
/* Set cpu mask to schedule testprogram2 on cpu 1 */
sched_buf.cpu = 2;
istat = sched_pgmadd(fbs_buf.fbs_id, &sched_buf);
printf("fbsid = %d¥n", fbs_buf.fbs_id);
if (istat != 0) {
   printf("could not schedule %s on cpu %d : errno = %d¥n",
   sched_buf.name_ptr, sched_buf.cpu>>1, errno);
   cleanup(fbs_buf.fbs_id);
   return(1);
}
pgm2_fpid = sched_buf.fpid;
printf("pgm2 fpid = %d¥n", pgm2_fpid);
/*
* SCHEDULE IDLE ON CPU 0
*
```

```
* The only parameter required for /idle is the CPU.
*/
sched_buf.name_ptr = "/idle";
sched_buf.cid = SCHED_OTHER; /* first-in-first out (FIFO) policy */
sched_buf.prior = 0;
sched_buf.param = 0; /* optional initiation parameter */
sched_buf.period = 1; /* time between wakeups */
sched_buf.cycle = 0; /* starting base cycle */
sched_buf.halt = 0; /* halt on overrun */
/* Set cpu mask to schedule idle on cpu 0 */
sched_buf.cpu = 1; /* mask = 1<<cpu */
istat = sched_pgmadd(fbs_buf.fbs_id, &sched_buf);
printf("fbsid = %d¥n", fbs_buf.fbs_id);
if (istat != 0) {
   printf("could not schedule %s on cpu %d : errno = %d¥n",
   sched_buf.name_ptr, sched_buf.cpu, errno);
   cleanup(fbs_buf.fbs_id);
   return(1);
}
idle0_fpid = sched_buf.fpid;
printf("idle0 fpid = %d¥n", idle0_fpid);
/*
* SCHEDULE IDLE ON CPU 1
*
* The only parameter required for /idle is the CPU.
*/
sched_buf.name_ptr = "/idle";
sched_buf.prior = 0;
sched_buf.cid = SCHED_OTHER; /* first-in-first out (FIFO) policy */
sched_buf.param = 0; /* optional initiation parameter */
sched_buf.period = 1; /* time between wakeups */
sched_buf.cycle = 0; /* starting base cycle */
sched_buf.halt = 0; /* halt on overrun */
/* Set cpu mask to schedule idle on cpu 1 */
sched_buf.cpu = 2; /* mask = 1<<cpu */
istat = sched_pgmadd(fbs_buf.fbs_id, &sched_buf);
if (istat != 0) {
   printf("could not schedule %s on cpu %d : errno = %d¥n",
   sched_buf.name_ptr, sched_buf.cpu, errno);
   cleanup(fbs_buf.fbs_id);
   return(1);
}
idle1_fpid = sched_buf.fpid;
printf("idle1 fpid = %d¥n", idle1_fpid);
/*
* ATTACH/SET REAL-TIME CLOCK
* Set the clock to interrupt every 10 msecs.
*/
count = 10000;
resolution = 1;
istat = fbsattach(fbs_buf.fbs_id, CLOCK1);
if (istat != 0) {
   printf("could not attach timing source: errno = %d¥n", errno);
   cleanup(fbs_buf.fbs_id);
   return(1);
}
```

```
istat = fbssetrtc(fbs_buf.fbs_id, count, resolution);
if (istat != 0) {
   printf("could not set rtc: errno = %d¥n", errno);
   cleanup(fbs_buf.fbs_id);
   return(1);
}
istat = fbsrunrtc(fbs_buf.fbs_id, START);
if (istat != 0) {
   printf("could not start rtc: errno = %d¥n", errno);
   cleanup(fbs_buf.fbs_id);
   return(1);
}
/*
* START PERFORMANCE MONITORING
*/
pmflg = 1;
/* zero out the "name" variable (fpid must be specified).
* Ulimately, the "name" variable can be used to store
* the full or relative path name of a test program.
*/
bzero(name, sizeof(name));
cpu = 0; /* not used if fpid is being used */
/* start performance monitoring for testprogram1 */
istat = pmprogram(fbs_buf.fbs_id, name, cpu, pgm1_fpid, pmflg);
if (istat != 0) {
   printf("could not start pm for testprogram1 : errno = %d¥n", errno);
   cleanup(fbs_buf.fbs_id);
   return(1);
}
/* start performance monitoring for testprogram2 */
istat = pmprogram(fbs_buf.fbs_id, name, cpu, pgm2_fpid, pmflg);
if (istat != 0) {
   printf("coul dnot start pm for testprogram2 : errno = %d¥n", errno);
   cleanup(fbs_buf.fbs_id);
   return(1);
}
/* start performance monitoring for idle on cpu 0 */
istat = pmprogram(fbs_buf.fbs_id, name, cpu, idle0_fpid, pmflg);
if (istat != 0) {
   printf("could not start pm for idle0 : errno = %d¥n", errno);
   cleanup(fbs_buf.fbs_id);
   return(1);
}
/* start performance monitoring for idle on cpu 1 */
istat = pmprogram(fbs_buf.fbs_id, name, cpu, idle1_fpid, pmflg);
if (istat != 0) {
   printf("could not start pm for idle1 : errno = %d¥n", errno);
   cleanup(fbs_buf.fbs_id);
   return(1);
}
/*
* START SCHEDULING
*/
intrflg = 1;
istat = fbsintrpt(fbs_buf.fbs_id, intrflg);
if (istat != 0) {
   printf("could not start scheduler : errno = %d¥n", errno);
```

```
   cleanup(fbs_buf.fbs_id);
   return(1);
}
/*
* QUERY PERFORMANCE MONITOR VALUES
* 1 second = 100 cycles 1 minute = 600 frames
* Query once per second for 1 minute
*/
pm_buf[0].fpid = pgm1_fpid; pgmname[0] = PROGRAM1;
pm_buf[1].fpid = pgm2_fpid; pgmname[1] = PROGRAM2;
pm_buf[2].fpid = idle0_fpid; pgmname[2] = IDLE0;
pm_buf[3].fpid = idle1_fpid; pgmname[3] = IDLE1;
sleep(1); /* sleep for a while */
printf("Please wait, performance information being gathered. ¥n");
cycle_buf.cframe = 0;
while ((cycle_buf.cframe < 600)) {
   istat = fbscycle(fbs_buf.fbs_id, &cycle_buf);
   if (istat != 0) {
      printf("could not query fbscycle: errno = %d¥n", errno);
      cleanup(fbs_buf.fbs_id);
      return(1);
   }
   istat = pmqrylist(fbs_buf.fbs_id, pm_buf, NUM_PROCS);
   if (istat != 0) {
      printf("could not query process: errno = %d¥n", errno);
      cleanup(fbs_buf.fbs_id);
      return(1);
   }
   fprintf(fp,"****************************************¥n");
   fprintf(fp,"¥tFRAME=%d, CYCLE=%d¥n", cycle_buf.cframe,
   cycle_buf.ccycle);
   fprintf(fp,"****************************************¥n¥n");
   /* Write performance information for each
   * process into a file
   */
   for (i = 0; i < NUM_PROCS; i++) {
      fprintf(fp,"pgm=%s, fpid=%d¥n", pgmname[i],
      pm_buf[i].fpid);
      fprintf(fp,"-------------------------------------¥n");
      fprintf(fp,"last cycle secs:nsecs = %d:%d¥n",
      pm_buf[i].lastcyc_tm.tv_sec,
      pm_buf[i].lastcyc_tm.tv_nsec);
      fprintf(fp,"total cycles = %d¥n",pm_buf[i].tot_cycles);
      fprintf(fp, "total cycles secs:nsecs = %d:%d¥n",
      pm_buf[i].tot_cycles_tm.tv_sec,
      pm_buf[i].tot_cycles_tm.tv_nsec);
      fprintf(fp, "overruns = %d¥n",pm_buf[i].overruns);
      fprintf(fp, "mincyc secs:nsecs = %d:%d¥n",
      pm_buf[i].mincyc_tm.tv_sec,
      pm_buf[i].mincyc_tm.tv_nsec);
      fprintf(fp, "mincyc_cycle = %d¥n",
      pm_buf[i].mincyc_cycle);
      fprintf(fp, "mincyc_frame = %d¥n",
      pm_buf[i].mincyc_frame);
      fprintf(fp, "maxcyc secs:nsecs = %d:%d¥n",
      pm_buf[i].maxcyc_tm.tv_sec,
```

```
            pm_buf[i].maxcyc_tm.tv_nsec);
            fprintf(fp, "maxcyc_cycle = %d¥n",
            pm_buf[i].maxcyc_cycle);
            fprintf(fp, "maxcyc_frame = %d¥n",
            pm_buf[i].maxcyc_frame);
            fprintf(fp, "minframe secs:nsecs = %d:%d¥n",
            pm_buf[i].minframe_tm.tv_sec,
            pm_buf[i].minframe_tm.tv_nsec);
            fprintf(fp, "minframe = %d¥n", pm_buf[i].minframe);
            fprintf(fp, "maxframe secs:nsecs = %d:%d¥n",
            pm_buf[i].maxframe_tm.tv_sec,
            pm_buf[i].maxframe_tm.tv_nsec);
            fprintf(fp, "maxframe = %d¥n", pm_buf[i].maxframe);
            fprintf(fp, "¥n");
         }
         sleep(1); /* sleep for a while */
      }
      printf("Performance data has been gathered. ¥n");
      if (istat != 0)
         printf("istat = %d¥n", istat);
      /* Stop PM on CPUs 0 and 1 */
      pmmonitor(fbs_buf.fbs_id, 0, 3);
      /* Stop the clock */
      istat = fbsrunrtc(fbs_buf.fbs_id, STOP);
      if (istat != 0) {
         printf("could not stop rtc: errno = %d¥n", errno);
         cleanup(fbs_buf.fbs_id);
         return(1);
      }
      /* Detach from the fbs */
      istat = fbsdetach(fbs_buf.fbs_id);
      if (istat != 0) {
         printf("could not dettach timing source: errno = %d¥n", errno);
         cleanup(fbs_buf.fbs_id);
         return(1);
      }
      /* Remove the fbs */
      istat = fbsremove(fbs_buf.fbs_id, -1);
      if (istat != 0) {
         printf("could not remove timing source: errno = %d¥n", errno);
         cleanup(fbs_buf.fbs_id);
         return(1);
      }
      cleanup(fbs_buf.fbs_id);
      return(0);
}
```

prog.c

```
#include <fbsched.h>
#include <time.h>

double fact(double x) {
    if (x == 1.0) return x;
    else return x * fact(x-1.0);
}

int main() {
    double factVal;
    struct timespec currTime;
    // Get/Create a seed for random number creation
    clock_gettime(CLOCK_REALTIME, &currTime);
    srandom( (unsigned) currTime.tv_nsec);

    while(1){
       fbswait( );
       factVal = fact ( (double) ((random( ) % 10 ) + 1) );
    }
}
```

Makefile

```
cc = gcc
FLAGS = -g -lccur_rt -lccur_fbsched

all: schedule prog1 prog2

schedule: schedule.c
    $(CC) $(FLAGS) -o schedule schedule.c

prog1: prog.c
    $(CC) $(FLAGS) -o prog1 prog.c
    cp prog1 prog2

clean:
    rm -f prog1 prog2 schedule pmresults *.*~
```

用語解説

デッドライン違反

FBS でスケジュールされたプロセスがデッドライン時間の終了前にその処理が終わらない時に発生する状況。デッドライン違反はスケジューラーを停止させる可能性があることに注意して下さい。

エンド・オブ・サイクル・スケジューリング

現在のメジャー・フレームの現在のマイナー・サイクルで実行するようにスケジュールされた最後のプロセスがその処理を終了した時、スケジューリングが起動される Frequency-Based Scheduler の方式。

フレーム・オーバーラン

FBS でスケジュールされたプロセスが再度実行をスケジュールされる前にその処理が終了しない場合に発生する状況。これはエンド・オブ・サイクル・スケジューリングではなく、割り込みスケジューリングにのみ適用されます。

頻度

Frequency-Based Scheduler が適用された場合は時間間隔を示します。この間隔は高分解能クロック、外部割込みソース、サイクルの終了に準ずることが可能です。

Frequency-Based Scheduler

プロセスをユーザー指定の頻度で実行できるようにする高分解能タスク同期化メカニズム。

ハード・オーバーラン

スケジュールされたプロセスの破滅的な障害であるフレーム・オーバーランの一種。

アイドル時間

CPU がビジーではない時間。

反復

Frequency-Based Scheduler によって起動されるプロセスの一つの段階。

最新時間

FBS でスケジュールされたプロセスが最後にスケジューラーに起動されてから **`fbswait(3)`** が呼ばれるまでに実行に費やした時間を示す Performance Monitor によって返される値。

メジャー・フレーム

Frequency-Based Scheduler が構成している全てのマイナー・サイクルを 1 回通り抜けること。メジャー・フレームはマイナー・サイクルの持続期間とメジャー・フレームあたりのマイナー・サイクルの数を掛けることで得られる持続期間に関連付けられています。

最大サイクル・サイクル

最大サイクル時間が発生したマイナー・サイクルの番号を示す Performance Monitor によって返される値。

最大サイクル・フレーム

最大サイクル時間が発生したメジャー・フレームの番号を示す Performance Monitor によって返される値。

最大サイクル時間

FBS でスケジュールされたプロセスがサイクルで実行に費やしたマイクロ秒単位の最大時間を示す Performance Monitor によって返される値。

最大フレーム・フレーム

最大フレーム時間が発生したメジャー・フレームの番号を示す Performance Monitor によって返される値。

最大フレーム時間

FBS でスケジュールされたプロセスがメジャー・フレーム中に実行に費やしたマイクロ秒単位の最大時間を示す Performance Monitor によって返される値。

最小サイクル・サイクル

最小サイクル時間が発生したマイナー・サイクルの番号を示す Performance Monitor によって返される値。

最小サイクル・フレーム

最小サイクル時間が発生したメジャー・フレームの番号を示す Performance Monitor によって返される値。

最小サイクル時間

FBS でスケジュールされたプロセスがサイクルで実行に費やしたマイクロ秒単位の最小時間を示す Performance Monitor によって返される値。

最小フレーム・フレーム

最小フレーム時間が発生したメジャー・フレームの番号を示す Performance Monitor によって返される値。

最小フレーム時間

FBS でスケジュールされたプロセスがメジャー・フレーム中に実行に費やしたマイクロ秒単位の最小時間を示す Performance Monitor によって返される値。

マイナー・サイクル

Frequency-Based Scheduler によって保持される頻度の最小単位。マイナー・サイクルはスケジューラーに接続されたタイミング・ソースによって生成される割り込みの合間を経過する時間である持続期間に関連付けられています。タイミング・ソースがリアルタイム・クロックである場合、マイナー・サイクルの持続期間はマイナー・サイクルあたりのクロック・カウント数とクロック・カウントあたりのマイクロ秒数を指定することで定義されます。

オーバーランの回数

FBS でスケジュールされたプロセスがフレーム・オーバーランを引き起こした回数を示す Performance Monitor によって返される値。

Performance Monitor

Frequency-Based Scheduler でスケジュールされたプロセスによる CPU の使用状況を監視することを可能にさせるメカニズム。

周期

どれくらいの頻度で指定したプログラムを各メジャー・フレームで起動するのかを指定する Frequency-Based Scheduler のパラメータ。頻度 1 はプログラムが各マイナー・サイクルで起動されることを示し、頻度 2 は 2 サイクル毎に起動されることを示します。

アクセス特権

プロセスが繊細な操作を実行する、またはシステムの制限を無効にすることが許されるメカニズム。

プロセス・ディスパッチ・レイテンシー

割り込みによって知らされる外部イベントの発生から、外部イベントを待っているプロセスが最初の命令をユーザー・モードで実行するまでに経過する時間。

スケジューラー・キー

Frequency-Based Scheduler 用にユーザーが指定した数値の識別子。

シールド・プロセッサー

割り込みやシステム・デーモンに関連する予測不可能な処理から保護されている高優先度タスクの実行に対して責任を負う CPU。

ソフト・オーバーラン

プロセスが許容されるソフト・オーバーランの数の上限に達した場合のみ破滅的な障害となるフレーム・オーバーランの一種。スケジュールされた各プロセスがソフト・オーバーランの上限を持っており、デフォルトは 0 となります。

一部のソフト・オーバーランはランダム、予測不可能、再発しそうにない外部イベントに起因します。他のソフト・オーバーランはマイナー・フレーム・オーバーランのみに起因します。ソフト・オーバーランはスケジュールされたプロセスにフレーム・オーバーランから復旧して同期化に戻る機会を与えます。

ソフト・オーバーランはエンド・オブ・サイクル・スケジューリングではなく、割り込みスケジューリングのみに適用します。

空き時間

次から構成されるプロセッサーの時間： (1) アイドル時間、(2) Frequency-Based Scheduler にスケジュールされていないプロセスの CPU 時間、(3) Performance Monitor が有効になっていない FBS でスケジュールされたプロセスの CPU 時間。

開始ベース・サイクル

FBS でスケジュールされたプロセスが各メジャー・フレームで起動される最初のマイナー・サイクルを指定する Frequency-Based Scheduler のスケジューリング・パラメータ。

タイミング・モード

Performance Monitor を実行するモード。割り込みを処理するのに費やす時間を Performance Monitor のタイミング値に含めるのか、それとも除外するのかを指定します。

合計反復回数

FBS でスケジュールされたプロセスがスケジューラーにより起動された回数を示す Performance Monitor によって返される値。

合計時間

FBS でスケジュールされたプロセスが全てのサイクルで実行するのに費やした合計時間を示す Performance Monitor によって返される値。

非シールド・プロセス

Frequency-Based Scheduler に起動されず **`fbswait`** を呼び出さない、つまり特定の頻度で実行するようにスケジュールされていないプロセス。

索引

A

ats コマンド 5-11
タイミング・ソースの接続　5-11, 6-8, 7-8

C

C ライブラリ呼出し手順
　Frequency-Based Scheduler　6-6
　Performance Monitor　6-70
C ライブラリ・ルーチン
　fbsaccess　6-7
　fbsattach　6-8
　fbsavail　6-9
　fbsconfigure　6-10
　fbscycle　6-13
　fbsdetach　6-14
　fbsdir　6-15
　fbsgetpid　6-16
　fbsgetrtc　6-17
　fbsid　6-18
　fbsinfo　6-19
　fbsinfo_big　6-19
　fbsintrpt　6-22
　fbsremove　6-23
　fbsresume　6-24
　fbsrunrtc　6-25
　fbsschedself　6-26
　fbssetrtc　6-28
　fbstrig　6-29
　fbswait　2-5, 6-30
　namepid　6-31
　namepid_big　6-31
　nametopid　6-31
　nametopid_big　6-31
　pgmremove　6-33
　pgmremove_big　6-33
　pgmtrigger　6-35
　pmclrpgm　6-71
　pmclrpgm_big　6-71
　pmclrtable　6-73
　pmclrtable_big　6-73
　pmmonitor　6-75
　pmmonitor_big　6-75
　pmprogram　6-77
　pmprogram_big　6-77
　pmqrycpu　6-79
　pmqrycpu_big　6-79
　pmqrylist　6-83
　pmqrypgm　6-85
　pmqrypgm_big　6-85
　pmqrytimer　6-88
　pmselect　6-89
　sched_fbsqry　6-36
　sched_fbsqry_big　6-36
　sched_pgm_deadline_query　6-39
　sched_pgm_deadline_query_big　6-39
　sched_pgm_deadline_test　6-42
　sched_pgm_deadline_test_big　6-42
　sched_pgm_set_deadline　6-45
　sched_pgm_set_deadline_big　6-45
　sched_pgm_set_soft_overrun_limit　6-48
　sched_pgm_set_soft_overrun_limit_big　6-48
　sched_pgmadd　6-51
　sched_pgmadd_args　6-54
　sched_pgmadd_args_big　6-54
　sched_pgmadd_attr　6-58
　sched_pgmadd_attr_big　6-58
　sched_pgmadd_big　6-51
　sched_pgmqry　6-61
　sched_pgmresched　6-65
　sched_pgmresched_big　6-65
C プログラムの実例　C-1
変更
　CPU バイアス　5-32, 6-51, 6-54, 6-58, 7-55, 7-57
　プロセス優先度　5-32, 6-51, 6-54, 6-58, 6-65, 7-55, 7-57
　スケジューラーのアクセス権　5-12, 6-7, 7-6
　スケジューリング・ポリシー　5-32, 6-51, 6-54, 6-58, 6-65, 7-55, 7-57
　タイミング・モード　5-42, 6-89, 7-81
chs コマンド　5-12
デッドライン時間の消去　5-32, 6-45, 7-48
Performance Monitor 値の消去　5-38, 6-71, 6-73, 7-67, 7-69
構成パラメータ　1-3
スケジューラーの構成　5-13, 6-10, 7-9
cpm コマンド　5-38
CPU バイアス　4-5, 6-51, 6-54, 6-58, 7-55, 7-57
cs コマンド 5-13

D

デッドライン
　時間の消去　5-32, 6-45, 7-48
　説明　2-5
　違反の検出 2-5
　問合せ　5-35, 6-39, 7-44
　時間の設定　5-32, 6-45, 7-48
　違反のテスト　6-42, 7-46
プロセスのデバッグ 2-7
タイミング・ソースの分離　5-15, 6-14, 7-12
ダイレクト・モード　5-2, 5-3
エンド・オブ・サイクル・スケジューリングの無効化
　5-15, 6-14, 7-12
分配された割り込み　3-3
dts コマンド　5-15

E

エッジ・トリガー割込み
　接続　5-11, 6-8, 7-8
　分離　5-15, 6-14, 7-12
　特殊デバイス・ファイル　3-3
　概要　3-3
　ユーザー・インターフェース　3-4
エンド・オブ・サイクル・スケジューリング
　定義　2-2
　無効化　5-15, 6-14, 7-12
　有効化　5-11, 6-8, 7-8
エラー・メッセージ　B-1
ex コマンド　5-47
実例
　C プログラム　C-1
　プロセスのスケジューリング　2-3, 2-4
　rtcp スクリプト　A-1
　シミュレーションの最適化　4-5
実行モード　5-2～5-4
rtcp の終了　5-47

F

fbsaccess　6-7, 7-6
fbsattach　6-8, 7-8
fbsavail　6-9
fbscheduler 1-4
fbsconfigure　6-10, 7-9
fbscycle　6-13, 7-11
fbsdetach　6-14, 7-12
fbsdir　6-15
fbsgetpid　6-16
fbsgetrtc　6-17, 7-13
fbsid　6-18, 7-14
fbsinfo　6-19, 7-15
fbsinfo_big　6-19
fbsintrpt　6-22, 7-17
fbsquery　7-18
fbsremove　6-23, 7-21
fbsresume　6-24, 7-22
fbsrunrtc　6-25, 7-24
fbsschedself　6-26, 7-25
fbssetrtc　6-28, 7-27
fbstrig　6-29
fbswait　2-5, 6-30, 7-28
FORTRAN ライブラリ呼出し手順
　Frequency-Based Scheduler　7-5
　Performance Monitor　7-66
FORTRAN ライブラリ・ルーチン
　fbsaccess　7-6
　fbsattach　7-8
　fbsconfigure　7-9
　fbscycle　7-11
　fbsdetach　7-12
　fbsgetrtc　7-13
　fbsid　7-14
　fbsinfo　7-15
　fbsintrpt　7-17
　fbsquery　7-18
　fbsremove　7-21
　fbsresume　7-22
　fbsrunrtc　7-24
　fbsschedself　7-25
　fbssetrtc　7-27
　fbswait　7-28
　nametopid　7-29
　pgmquery　7-30
　pgmremove　7-32
　pgmreschedule　7-34
　pgmschedule　7-37
　pgmstat　7-40
　pgmtrigger　7-42
　pmclrpgm　7-67
　pmclrtable　7-69
　pmmonitor　7-70
　pmprogram　7-71
　pmqrycpu　7-73
　pmqrylist　7-75
　pmqrypgm　7-77
　pmquerytimer　7-80
　pmselect　7-81
　sched_pgm_deadline_query　7-44
　sched_pgm_deadline_test　7-46
　sched_pgm_set_deadline　7-48
　sched_pgm_set_soft_overrun_limit　7-50
　sched_pgm_soft_overrun_query　7-51
　schedfbsqry　7-52
　schedpgmadd　7-55

schedpgmadd_args 7-57
schedpgmqry 7-59
schedpgmresched 7-62
フレーム・オーバーラン、オーバーランを参照
頻度 2-2
Frequency-Based Scheduler
C ライブラリ呼出し手順 6-6
C ライブラリ・ルーチン 6-4, 6-7
構成 1-3
FORTRAN ライブラリ呼出し手順 7-5
FORTRAN ライブラリ・ルーチン 7-6
概要 1-1, 2-1
アクセス特権 1-4
rtcp コマンドの手順 5-9
rtcp コマンド 5-7, 5-8, 5-10
タイミング・ソース、タイミング・ソース・ユーザー・インターフェースの参照 2-7

G

用語解説 Glossary-1
gtc コマンド 5-23

H

ハード・オーバーラン、オーバーランを参照
he コマンド 5-47
高分解能プロセス・アカウンティング 1-3

I

アイドル時間 4-3, 4-6, 6-26
対話モード 5-4
割り込み時間の包含/除外 4-1, 5-42, 6-88, 6-89, 7-80, 7-81
反復 4-1, 4-2

L

最新時間 4-2
システム上の全スケジューラーのリスト 5-21
負荷の平均化 4-5
ls コマンド 5-21

M

メジャー・フレーム 2-4, 4-1, 5-18, 6-13, 7-11
マニュアルの構成 iii
最大
サイクル・サイクル 4-2, 5-43, 6-79, 6-83, 7-73, 7-75
サイクル・フレーム 4-2, 5-43, 6-79, 6-83, 7-73, 7-75
サイクル時間 4-2, 5-43, 6-79, 6-83, 7-73, 7-75
フレーム・フレーム 4-2, 5-43, 6-79, 6-83, 7-73, 7-75
フレーム時間 4-2, 5-43, 6-79, 6-83, 7-73, 7-75
最小
サイクル・サイクル 4-2, 5-43, 6-79, 6-83, 7-73, 7-75
サイクル・フレーム 4-2, 5-43, 6-79, 6-83, 7-73, 7-75
サイクル時間 4-2, 5-43, 6-79, 6-83, 7-73, 7-75
フレーム・フレーム 4-2, 5-43, 6-79, 6-83, 7-73, 7-75
フレーム時間 4-2, 5-43, 6-79, 6-83, 7-73, 7-75
マイナー・サイクル 2-2, 4-2, 5-18, 6-13, 7-11
変更
CPU バイアス 5-32, 6-51, 6-54, 6-58, 7-55, 7-57
プロセス優先度 5-32, 6-51, 6-54, 6-58, 6-65, 7-55, 7-57
スケジューラーのアクセス権 5-12, 6-7, 7-6
スケジューリング・ポリシー 5-32, 6-51, 6-54, 6-58, 6-65, 7-55, 7-57
タイミング・モード 5-42, 6-89, 7-81

N

namepid 6-31
namepid_big 6-31
nametopid 2-7, 6-31, 7-29
nametopid_big 6-31
NightSim iii, 1-1
NightView 2-7

O

取得
デッドラインの状況 6-39, 7-44
マイナー・サイクル/メジャー・フレームの総数 5-18, 6-13, 7-11
Performance Monitor の値 5-43, 6-79, 6-83, 6-85, 7-73, 7-75, 7-77
プロセス ID 6-16, 6-31, 7-29
プロセスの情報 5-35, 6-36, 6-61, 7-18, 7-30, 7-40, 7-52, 7-59
プロセスの初期化パラメータ 7-43
リアルタイム・クロックの設定 5-23, 6-17, 7-13
キーに対するスケジューラー識別子 6-18, 7-14
スケジューラーの情報 5-19, 6-10, 6-18, 6-19, 7-9, 7-14, 7-15
スケジューラー・キーのリスト 6-15
ソフト・オーバーランの状況 5-35, 7-51
タイミング・モード 5-42, 6-88, 7-80
オーバーラン
定義 2-4, 4-2, 6-30, 7-28
ソフト・オーバーラン上限の設定 5-32, 6-48, 7-50
ソフト・オーバーランの状況 5-35, 7-51

P

PAM　1-4
pam_capability　1-4
Performance Monitor
　C ライブラリ呼出し手順　6-70
　C ライブラリ・ルーチン　6-69, 6-70
　構成　1-3
　FORTRAN ライブラリ呼出し手順　7-66
　FORTRAN ライブラリ・ルーチン　7-65, 6-67
　概要　1-2, 4-1
　rtcp コマンドの手順　5-10
　rtcp コマンド　5-8
　シミュレーションの実例　4-5
　開始/停止　5-40, 6-75, 6-77, 7-70, 7-71
　タイミング・モード　4-1, 6-88, 6-89, 7-80, 7-81
　ユーザー・インターフェース　4-5
　値
　　消去　5-38, 6-71, 6-73, 7-67, 7-69
　　定義　4-1, 4-2
　　取得　5-43, 6-79, 6-83, 6-85, 7-73, 7-75, 7-77
周期　2-2
アクセス権
　FBS　1-4
　スケジューラー　5-12, 5-13, 6-7, 7-6
pgmquery　7-30
pgmremove　6-33, 7-32
pgmremove_big　6-33
pgmreschedule　7-34
pgmschedule　7-37
pgmstat　7-40
pgmtrigger　6-35, 7-42
Pluggable Authentication Module (PAM)　1-4
pm コマンド　5-40
pmclrpgm　6-71, 7-67
pmclrpgm_big　6-71
pmclrtable　6-73, 7-69
pmclrtable_big　6-73
pmmonitor　6-75, 7-70
pmprogram　6-77, 7-71
pmprogram_big　6-77
pmqrycpu　6-79, 7-73
pmqrycpu_big　6-79
pmqrylist　6-83, 7-75
pmqrypgm　6-85, 7-77
pmqrypgm_big　6-85
pmqrytimer　7-80
pmselect　6-89, 7-81
アクセス特権　アクセス権を参照
プロセスを再スケジュール　5-28, 6-65, 7-34, 7-62
プロセスをスケジュール　2-2, 5-32, 6-26, 6-51, 6-54, 6-58, 7-25, 7-37, 7-55, 7-57
プロセスをスリープ　6-30, 7-28
問合せ
　デッドラインの状況　6-39, 7-44
　FBS カーネル構成　6-9
　Performance Monitor の値　5-43, 6-79, 6-83, 6-85, 7-73, 7-75, 7-77
　プロセス情報　5-35, 6-36, 6-61, 7-18, 7-30, 7-40, 7-52, 7-59
　リアルタイム・クロック　5-23, 6-17, 7-13
　スケジュール済みプロセス　5-35, 6-36, 6-39, 6-61, 7-18, 7-40, 7-52, 7-59
　スケジューラー構成　5-19, 6-10, 7-9
　ソフト・オーバーラン処理　5-35, 7-51
　タイミング・モード　5-42, 6-88, 7-80

R

rc コマンド　5-21
RCIM
　デバイス・ファイル　1-4
　カーネル・パラメータ　1-3
　タイミング・デバイス　1-2, 3-3
リアルタイム・クロック
　接続　5-1, 6-8, 7-8
　C ルーチンの要約　3-3
　分離　5-15, 6-14, 7-12
　デバイス特殊ファイル　3-2
　設定の取得　5-23, 6-17, 7-13
　概要　3-1
　手順　3-2
　rtcp コマンドの要約　3-3
　設定　5-22, 6-28, 7-27
　開始　5-21, 6-25, 7-24
　停止　5-22, 6-25, 7-24
　ユーザー・インターフェース　3-2
関連図書　iv
削除
　プロセスをスケジューラーから　5-26, 6-33, 7-32
　スケジューラー　5-16, 6-23, 7-21
プロセスを再スケジュール　5-28, 6-65, 7-34, 7-62
resume コマンド 5-25
FBS の再開　5-25, 6-22, 6-24, 7-17, 7-22
rmp コマンド　5-26
rms コマンド　5-16
rsp コマンド　5-28
rtcp
　コマンド・スクリプト　5-3, A-1
　コマンドの手順
　　FBS　5-9
　　PM　5-10
　コマンドの要約　5-7

コマンド
ats 5-11
chs 5-12
cpm 5-38
cs 5-13
dts 5-15
ex 5-47
gtc 5-23
he 5-47
ls 5-21
pm 5-40
rc 5-21
resume 5-25
rmp 5-26
rms 5-16
rsp 5-28
sc 5-22
sp 5-32
start 5-24
stc 5-22
stop 5-25
svs 5-17
vc 5-18
vcm 5-42
vp 5-35
vpm 5-43
vs 5-19
エラー B-1
スクリプトの実例 A-1
実行モード 5-2～5-4
終了 5-47
ヘルプ 5-5, 5-47
概要 5-1
rtparm 7-43

S

スケジューラー構成の保存 5-3, 5-17
sc コマンド 5-22
sched_fbsqry 6-36
sched_fbsqry_big 6-36
sched_pgm_deadline_query 6-39, 7-44
sched_pgm_deadline_query_big 6-39
sched_pgm_deadline_test 6-42, 7-46
sched_pgm_deadline_test_big 6-42
sched_pgm_set_deadline 6-45, 7-48
sched_pgm_set_deadline_big 6-45
sched_pgm_set_soft_overrun_limit 6-48, 7-50
sched_pgm_set_soft_overrun_limit_big 6-48
sched_pgm_soft_overrun_query 7-51
sched_pgmadd 6-51
sched_pgmadd_args 6-54
sched_pgmadd_args_big 6-54
sched_pgmadd_attr 6-58
sched_pgmadd_attr_big 6-58
sched_pgmadd_big 6-51
sched_pgmqry 6-61
sched_pgmresched 6-65
sched_pgmresched_big 6-65
schedfbsqry 7-52
schedpgmadd 7-55
schedpgmadd_args 7-57
schedpgmqry 7-59
schedpgmresched 7-62
プロセスをスケジュール 2-2, 5-32, 6-26, 6-51, 6-54, 6-58, 7-25, 7-37, 7-55, 7-57
スケジューラーの構成 5-13, 6-10, 7-9
スケジューラーの頻度 2-2
リアルタイム・クロックの設定 5-22, 6-28, 7-27
デッドライン時間の設定 6-45, 7-48
ソフト・オーバーラン上限の設定 5-32, 6-48, 7-50
ソフト・オーバーラン オーバーランを参照
ソフトウェア要求 1-2
sp コマンド 5-32
空き時間 4-3, 4-6
開始
Frequency-Based Scheduler 5-24, 6-22, 7-17
Performance Monitor 5-40, 6-75, 6-77, 7-70, 7-71
リアルタイム・クロック 5-21, 6-25, 7-24
start コマンド 5-24
開始ベース・サイクル 2-2
stc コマンド 5-22
停止
Frequency-Based Scheduler 5-25, 6-22, 7-17
Performance Monitor 5-40, 6-75, 6-77, 7-70, 7-71
リアルタイム・クロック 5-22, 6-25, 7-24
stop コマンド 5-25
svs コマンド 5-3
構文記法 iv

T

デッドライン違反のテスト 6-42, 7-46
タイミング・モード 4-1, 5-42, 6-88, 6-89, 7-80, 7-81
タイミング・ソース
接続 5-11, 6-8, 7-8
分離 5-15, 6-14, 7-12
概要 3-1
特権 1-4
合計反復数 4-2
合計時間 4-2
プロセスの起動 6-29, 6-35, 7-42

U

非シールド・プロセス　4-4

V

vc コマンド　5-18
vcm コマンド　5-42
vp コマンド　5-35
vpm コマンド　5-43
vs コマンド　5-19

W

Frequency-Based Scheduler を待機　6-30, 7-28
スリープ中のプロセスを起動　6-29, 6-35, 7-42