リアルタイム API トレーニングテキスト

ベーシックコース

2009/12/01 第3版
コンカレント日本株式会社
プロフェッショナルサービス部

ANSI-C 規格

C 言語規格であるが、UNIX の概念を取り込んでいる、ただしこの規格にはプロセスの概念はない
例

raise − send signal to program

リアルタイム処理に重要な宣言文

sig_atomic_t　　アトミック操作で参照可能なデータ型
変数への書き込みは割り込まれない

volatile　　メモリを必ずアクセスする(レジスタやキャッシュされない)データ型
シグナルハンドラでは、sig_atomic_t とペアで用いられる
実デバイスを扱う場合には、必須の型規定子

例

```
volatile sigatomic_t atomic_value;
typedef struct
{
        union
        {
        volatile unsigned char          mem08[4096];
        volatile unsigned short int     mem16[2048];
        volatile unsigned long int      mem32[1024];
        volatile unsigned long long int mem64[ 512];
        } access;
} USERDEVICE;
```

マルチプロセッサ間で、変数を共有する場合には必須

追加の規格
C9X Extensions

```
//
unsigned long long [int] long_long_int;
long_long_int = 0LL;
printf("val %lld",long_lobg_int);
```

ANSI C Extensions

```
long float == double
long double                     8 バイト double
```

下記に示すように 32bit 版および 32bit エミュレーションと 64bit 版では、データサイズが異なる。

データサイズを調べるプログラム	32bit 版及び 32bit エミュレーションの場合	64bit 版の場合

main()
{
int x;
 	
printf("char %d¥n",sizeof(char));
printf("short %d¥n",sizeof(short));
printf("long %d¥n",sizeof(long));
printf("void* %d¥n",sizeof(void*));
printf("int %d¥n",sizeof(int));
printf("unsigned int %d¥n",sizeof(unsigned int));
printf("unsigned long int %d¥n",sizeof(unsigned long int));
printf("long int %d¥n",sizeof(long int));
printf("long long %d¥n",sizeof(long long));
printf("long long int %d¥n",sizeof(long long int));
printf("x %X %d¥n",&x,sizeof(&x));
}
	char 1
	short 2
	long 4
	void* 4
	int 4
	unsigned int 4
	unsigned long int 4
	long int 4
	long long 8
	long long int 8
	x FFFFD780 4
		char 1
		short 2
		long 8
		void* 8
		int 4
		unsigned int 4
		unsigned long int 8
		long int 8
		long long 8
		long long int 8
		x FFFFE5FC 8

POSIX 標準と Linux

IEEE POSIX1003.1 標準システムコール規格(ANSI-C+UNIX システムコール)
IEEE POSIX1003.1b リアルタイム規格
リアルタイムに必要な機能をプロセス(タスク)の概念で規格化したもの
IEEE POSIX1003.1c スレッド規格
1b とは無関係にスレッドを規格化したもの

Linuxの POSIX 標準 (以下の情報は POSIX.1-2001 の抜粋) は互換システムの動作とインタフェースのセットを記述している。しかし、多くのインタフェースは選択可能であり、コンパイル時にインタフェースが使用可能かをテストする機能テストマクロと、実行時にテストする関数 sysconf(3), fpathconf(3),pathconf(3),confstr(3) がある。シェルスクリプトでは getconf(1) を使うことができる。詳細は sysconf(3) を参照すること。

POSIX 省略形の名前・オプション・オプションを調べるための sysconf() 引き数の名前・(可能ならば) 非常に短い説明を記述する。より正確な詳細は POSIX 標準自身に書かれている。 POSIX 標準は今日では Web で自由にアクセスできる。

ADV - _POSIX_ADVISORY_INFO - _SC_ADVISORY_INFO 200112
アドバイスの関数
posix_fadvise(),
posix_fallocate(),
posix_memalign(),
posix_madvise()
が存在する。

AIO - _POSIX_ASYNCHRONOUS_IO - _SC_ASYNCHRONOUS_IO 200112
ヘッダ <aio.h> が存在する。関数
aio_cancel(),
aio_error(),
aio_fsync(),
aio_read(),
aio_return(),
aio_suspend(),
aio_write(),
lio_listio()
が存在する。

BAR - _POSIX_BARRIERS - _SC_BARRIERS 200112
このオプションは _POSIX_THREADS と _POSIX_THREAD_SAFE_FUNCTIONS オプションを暗黙の内に指定する。
関数
pthread_barrier_destroy(),
pthread_barrier_init(),
pthread_barrier_wait(),
pthread_barrierattr_destroy(),
pthread_barrierattr_init()
が存在する。

—— - POSIX_CHOWN_RESTRICTED 1
このオプションが有効な場合 (POSIX.1-2001 では常に有効)、 root だけが ファイル所有者の変更を許され、root 以外はファイルのグループを自分が所属するグループの 1 つに設定することだけができる。これは以下の関数に影響する。 chown(), fchown().

CS - _POSIX_CLOCK_SELECTION - _SC_CLOCK_SELECTION 200112

このオプションは _POSIX_TIMERS オプションを暗黙の内に指定する。関数
pthread_condattr_getclock(),
pthread_condattr_setclock(),
clock_nanosleep()
が存在する。 CLOCK_REALTIME が関数 clock_settime() で変更された場合、絶対時間に関係する全てのタイマのセットに影響する。

CPT – _POSIX_CPUTIME – _SC_CPUTIME **200112**
clockID CLOCK_PROCESS_CPUTIME_ID がサポートされている。このクロックの初期値は、各プロセス毎に 0 となる。このオプションは _POSIX_TIMERS オプションを暗黙の内に指定する。関数 clock_getcpuclockid() が存在する。

—— – _POSIX_FILE_LOCKING – _SC_FILE_LOCKING **未定義**
このオプションは削除された。XPG6 最終版にはない。

FSC – _POSIX_FSYNC – _SC_FSYNC **200112**
関数 fsync() が存在する。

IP6 – _POSIX_IPV6 – _SC_IPV6 **200112**
Internet Protocol Version 6 がサポートされている。

—— – _POSIX_JOB_CONTROL – _SC_JOB_CONTROL **1**
このオプションが有効な場合 (POSIX.1-2001 では常に有効)、シ ス テ ム は POSIX 方式のジョブ制御を実装しており、関数
setpgid(),
tcdrain(),
tcflush(),
tcgetpgrp(),
tcsendbreak(),
tcsetattr(),
tcsetpgrp()
が存在する。

MF – _POSIX_MAPPED_FILES – _SC_MAPPED_FILES **200112**
共有メモリがサポートされている。インクルードファイル <sys/mman.h> が存在する。関数 mmap(), msync(), munmap() が存在する。

ML – _POSIX_MEMLOCK – _SC_MEMLOCK **200112**
共有メモリがコア内にロックできる。関数 mlockall(), munlockall() が存 在する。

MR/MLR – _POSIX_MEMLOCK_RANGE – _SC_MEMLOCK_RANGE **200112**
よ り詳細に、範囲をコア内にロックできる。関数 mlock(), munlock() が存在する。

MPR – _POSIX_MEMORY_PROTECTION – _SC_MEMORY_PROTECTION **200112**
関数 mprotect() が存在する。

MSG – _POSIX_MESSAGE_PASSING – _SC_MESSAGE_PASSING **200112**
インクルードファイル <mqueue.h> が存在する。関数
mq_close(),
mq_getattr(),
mq_notify(),
mq_open(),
mq_receive(),
mq_send(),

mq_setattr(),
mq_unlink()
が存在する。

MON – _POSIX_MONOTONIC_CLOCK – _SC_MONOTONIC_CLOCK 200112
CLOCK_MONOTONIC がサポートされている。
このオプションは _POSIX_TIMERS オプションを暗黙の内に指定する。
影響を受ける関数は以下の通り。
aio_suspend(),
clock_getres(),
clock_gettime(),
clock_settime(),
timer_create().

—— – _POSIX_MULTI_PROCESS – _SC_MULTI_PROCESS 未定義
このオプションは削除された。XPG6 最終版にはない。

—— – _POSIX_NO_TRUNC 1
このオプションが有効な場合 (POSIX.1-2001 では常に有効)、 NAME_MAX よ り長 いパス名の構成要素は切り詰められないが、エラーになる。この設定は構成要素のパス接頭辞に依存する場合もある。

PIO – _POSIX_PRIORITIZED_IO – _SC_PRIORITIZED_IO 200112
このオプションは非同期 I/O の優先度が指定できることを表す。これは以下の関数に影響する。
aio_read(),
aio_write().

PS – _POSIX_PRIORITY_SCHEDULING – _SC_PRIORITY_SCHEDULING 200112
インクルードファイル <sched.h> が存在する。関数
sched_get_priority_max(),
sched_get_priority_min(),
sched_getparam(),
sched_getscheduler(),
sched_rr_get_interval(),
sched_setparam(),
sched_setscheduler(),
sched_yield()
が存在する。 _POSIX_SPAWN も有効な場合は、関数 200112
posix_spawnattr_getschedparam(),
posix_spawnattr_getschedpolicy(),
posix_spawnattr_setschedparam(),
posix_spawnattr_setschedpolicy()
が存在する。

RS – _POSIX_RAW_SOCKETS 200112
raw ソケットがサポートされている。影響を受ける関数は以下の通り。
getsockopt(), setsockopt().

—— – _POSIX_READER_WRITER_LOCKS – _SC_READER_WRITER_LOCKS 200112
このオプションは _POSIX_THREADS オプションを暗黙の内に指定する 。 逆 に POSIX.1-2001 では _POSIX_THREADS オプションはこのオプションを暗黙の内に指定する。
関数

pthread_rwlock_destroy(),
pthread_rwlock_init(),
pthread_rwlock_rdlock(),
pthread_rwlock_tryrdlock(),
pthread_rwlock_trywrlock(),
pthread_rwlock_unlock(),
pthread_rwlock_wrlock(),
pthread_rwlockattr_destroy(),
pthread_rwlockattr_init()
が存在する。

RTS – _POSIX_REALTIME_SIGNALS – _SC_REALTIME_SIGNALS 200112
リアルタイムシグナルがサポートされている。関数
sigqueue(),
sigtimedwait(),
sigwaitinfo()
が存在する。

–– – _POSIX_REGEXP – _SC_REGEXP 1
このオプションが有効な場合 (POSIX.1-2001 では常に有効)、 POSIX 正規表現がサポートされ、関数
regcomp(),
regerror(),
regexec(),
regfree()
が存在する。

–– – _POSIX_SAVED_IDS – _SC_SAVED_IDS 1
こ のオプションが有効な場合 (POSIX.1-2001 では常に有効)、プロセスは保存 (saved) set-user-ID と保存 set-group-ID を持つ。影響を受ける関数は以 下の通り。
exec(),
kill(),
seteuid(),
setegid(),
setgid(),
setuid().

SEM – _POSIX_SEMAPHORES – _SC_SEMAPHORES 200112
インクルードファイル <semaphore.h> が存在する。関数
sem_close(),
sem_destroy(),
sem_getvalue(),
sem_init(),
sem_open(),
sem_post(),
sem_trywait(),
sem_unlink(),
sem_wait()
が存在する。

SHM – _POSIX_SHARED_MEMORY_OBJECTS – _SC_SHARED_MEMORY_OBJECTS 200112
関数
mmap(),

munmap(),
shm_open(),
shm_unlink()
が存在する。

── - _POSIX_SHELL - _SC_SHELL 1
こ の オプションが有効な場合 (POSIX.1-2001 では常に有効)、関数 system()
が存在する。

SPN - _POSIX_SPAWN - _SC_SPAWN 200112
このオプションは、例えば MMU が存在しないなどの理由によって、 fork() を
使 用することが難しいか不可能という状況で、プロセス生成をサポートするこ
とを表す。 _POSIX_SPAWN が有効な場合、インクルードファイル <spawn.h> と
関数
posix_spawn(),
posix_spawn_file_actions_addclose(),
posix_spawn_file_actions_adddup2(),
posix_spawn_file_actions_addopen(),
posix_spawn_file_actions_destroy(),
posix_spawn_file_actions_init(),
posix_spawnattr_destroy(),
posix_spawnattr_getsigdefault(),
posix_spawnattr_getflags(),
posix_spawnattr_getpgroup(),
posix_spawnattr_getsigmask(),
posix_spawnattr_init(),
posix_spawnattr_setsigdefault(),
posix_spawnattr_setflags(),
posix_spawnattr_setpgroup(),
posix_spawnattr_setsigmask(),
posix_spawnp()
が存在する。 _POSIX_PRIORITY_SCHEDULING も有効な場合、関数 200112
posix_spawnattr_getschedparam(),
posix_spawnattr_getschedpolicy(),
posix_spawnattr_setschedparam(),
posix_spawnattr_setschedpolicy()
が存在する。

SPI - _POSIX_SPIN_LOCKS - _SC_SPIN_LOCKS 200112
こ のオプションは _POSIX_THREADS と _POSIX_THREAD_SAFE_FUNCTIONS オプシ
ョンを暗黙の内に指定する。関数
pthread_spin_destroy(),
pthread_spin_init(),
pthread_spin_lock(),
pthread_spin_trylock(),
pthread_spin_unlock()
が存在する。

SS - _POSIX_SPORADIC_SERVER - _SC_SPORADIC_SERVER 未定義
スケジューリングポリシー SCHED_SPORADIC がサポートされている。このオ プ
シ ョンは _POSIX_PRIORITY_SCHEDULING オプションを暗黙の内に指定する。影
響を受ける関数は以下の通り。
sched_setparam(),
sched_setscheduler().

SIO – _POSIX_SYNCHRONIZED_IO – _SC_SYNCHRONIZED_IO 200112
影響を受ける関数は以下の通り。 open(), msync(), fsync(), fdatasync().

TSA – _POSIX_THREAD_ATTR_STACKADDR – _SC_THREAD_ATTR_STACKADDR 200112
影響を受ける関数は以下の通り。
pthread_attr_getstack(),
pthread_attr_getstackaddr(),
pthread_attr_setstack(),
pthread_attr_setstackaddr().

TSS – _POSIX_THREAD_ATTR_STACKSIZE – _SC_THREAD_ATTR_STACKSIZE 200112
影響を受ける関数は以下の通り。
pthread_attr_getstack(),
pthread_attr_getstacksize(),
pthread_attr_setstack(),
pthread_attr_setstacksize().

TCT – _POSIX_THREAD_CPUTIME – _SC_THREAD_CPUTIME 200112
clockID CLOCK_THREAD_CPUTIME_ID がサポートされている。このオプション は _POSIX_TIMERS オプションを暗黙の内に指定する。影響を受ける関数は以下の通り。
pthread_getcpuclockid(),
clock_getres(),
clock_gettime(),
clock_settime(),
timer_create().

TPI – _POSIX_THREAD_PRIO_INHERIT – _SC_THREAD_PRIO_INHERIT
200112
影響を受ける関数は以下の通り。
pthread_mutexattr_getprotocol(),
pthread_mutexattr_setprotocol().

TPP – _POSIX_THREAD_PRIO_PROTECT – _SC_THREAD_PRIO_PROTECT 200112
影響を受ける関数は以下の通り。
pthread_mutex_getprioceiling(),
pthread_mutex_setprioceiling(),
pthread_mutexattr_getprioceiling(),
pthread_mutexattr_getprotocol(),
pthread_mutexattr_setprioceiling(),
pthread_mutexattr_setprotocol().

TPS – _POSIX_THREAD_PRIORITY_SCHEDULING – _SC_THREAD_PRIORITY_SCHEDULING 200112
このオプションが有効な場合、1 つのプロセス内の個々のスレッドを個々の 優先 度または個々のスケジューラ (またはその両方) で実行できる。影響を受ける関数は以下の通り。
pthread_attr_getinheritsched(),
pthread_attr_getschedpolicy(),
pthread_attr_getscope(),
pthread_attr_setinheritsched(),
pthread_attr_setschedpolicy(),
pthread_attr_setscope(),
pthread_getschedparam(),

pthread_setschedparam(),
pthread_setschedprio().

TSH – _POSIX_THREAD_PROCESS_SHARED – _SC_THREAD_PROCESS_SHARED **200112**

影響を受ける関数は以下の通り。

pthread_barrierattr_getpshared(),
pthread_barrierattr_setpshared(),
pthread_condattr_getpshared(),
pthread_condattr_setpshared(),
pthread_mutexattr_getpshared(),
pthread_mutexattr_setpshared(),
pthread_rwlockattr_getpshared(),
pthread_rwlockattr_setpshared().

TSF – _POSIX_THREAD_SAFE_FUNCTIONS – _SC_THREAD_SAFE_FUNCTIONS **200112**

影響を受ける関数は以下の通り。

readdir_r(),
getgrgid_r(),
getgrnam_r(),
getpwnam_r(),
getpwuid_r(),
flockfile(),
ftrylockfile(),
funlockfile(),
getc_unlocked(),
getchar_unlocked(),
putc_unlocked(),
putchar_unlocked(),
rand_r(),
strerror_r(),
strtok_r(),
asctime_r(),
ctime_r(),
gmtime_r(),
localtime_r().

TSP – _POSIX_THREAD_SPORADIC_SERVER – _SC_THREAD_SPORADIC_SERVER **未定義**

このオプションは _POSIX_THREAD_PRIORITY_SCHEDULING オプションを暗黙の内に指定する。影響を受ける関数は以下の通り。

sched_getparam(),
sched_setparam(),
sched_setscheduler().

THR – _POSIX_THREADS – _SC_THREADS **200112**

POSIX スレッドの基本サポートが使用可能である。関数

pthread_atfork(),
pthread_attr_destroy(),
pthread_attr_getdetachstate(),
pthread_attr_getschedparam(),
pthread_attr_init(),
pthread_attr_setdetachstate(),
pthread_attr_setschedparam(),
pthread_cancel(),
pthread_cleanup_push(),

pthread_cleanup_pop(),
pthread_cond_broadcast(),
pthread_cond_destroy(),
pthread_cond_init(),
pthread_cond_signal(),
pthread_cond_timedwait(),
pthread_cond_wait(),
pthread_condattr_destroy(),
pthread_condattr_init(),
pthread_create(),
pthread_detach(),
pthread_equal(),
pthread_exit(),
pthread_getspecific(),
pthread_join(,
pthread_key_create(),
pthread_key_delete(),
pthread_mutex_destroy(),
pthread_mutex_init(),
pthread_mutex_lock(),
pthread_mutex_trylock(),
pthread_mutex_unlock(),
pthread_mutexattr_destroy(),
pthread_mutexattr_init(),
pthread_once(),
pthread_rwlock_destroy(),
pthread_rwlock_init(),
pthread_rwlock_rdlock(),
pthread_rwlock_tryrdlock(),
pthread_rwlock_trywrlock(),
pthread_rwlock_unlock(),
pthread_rwlock_wrlock(),
pthread_rwlockattr_destroy(),
pthread_rwlockattr_init(),
pthread_self(),
pthread_setcancelstate(),
pthread_setcanceltype(),
pthread_setspecific(),
pthread_testcancel()

が存在する。

TMO – _POSIX_TIMEOUTS – _SC_TIMEOUTS **200112**

関数

mq_timedreceive(),
mq_timedsend(),
pthread_mutex_timedlock(),
pthread_rwlock_timedrdlock(),
pthread_rwlock_timedwrlock(),
sem_timedwait(),
posix_trace_timedgetnext_event()

が存在する。

TMR – _POSIX_TIMERS – _SC_TIMERS **200112**

関数

clock_getres(),
clock_gettime(),
clock_settime(),
nanosleep(),
timer_create(),
timer_delete(),
timer_gettime(),
timer_getoverrun(),
timer_settime()
が存在する。

TRC – _POSIX_TRACE – _SC_TRACE 未定義

POSIX トレーシング（tracing）が使用可能である。関数
posix_trace_attr_destroy(),
posix_trace_attr_getclockres(),
posix_trace_attr_getcreatetime(),
posix_trace_attr_getgenversion(),
posix_trace_attr_getmaxdatasize(),
posix_trace_attr_getmaxsystemeventsize(),
posix_trace_attr_getmaxuserеventsize(),
posix_trace_attr_getname(),
posix_trace_attr_getstreamfullpolicy(),
posix_trace_attr_getstreamsize(),
posix_trace_attr_init(),
posix_trace_attr_setmaxdatasize(),
posix_trace_attr_setname(),
posix_trace_attr_setstreamsize(),
posix_trace_attr_setstreamfullpolicy(),
posix_trace_clear(),
posix_trace_create(),
posix_trace_event(),
posix_trace_eventid_equal(),
posix_trace_eventid_get_name(),
posix_trace_eventid_open(),
posix_trace_eventtypelist_getnext_id(),
posix_trace_eventtypelist_rewind(),
posix_trace_flush(),
posix_trace_get_attr(),
posix_trace_get_status(),
posix_trace_getnext_event(),
posix_trace_shutdown(),
posix_trace_start(),
posix_trace_stop(),
posix_trace_trygetnext_event()
が存在する。

TEF – _POSIX_TRACE_EVENT_FILTER – _SC_TRACE_EVENT_FILTER 未定義

このオプションは _POSIX_TRACE オプションを暗黙の内に指定する。関数
posix_trace_eventset_add(),
posix_trace_eventset_del(),
posix_trace_eventset_empty(),
posix_trace_eventset_fill(),
posix_trace_eventset_ismember(),
posix_trace_get_filter(),

posix_trace_set_filter(),
posix_trace_trid_eventid_open()
が存在する。

TRI – _POSIX_TRACE_INHERIT – _SC_TRACE_INHERIT **未定義**

ト レースされているプロセスの子プロセスのトレースをサポートする。このオプションは _POSIX_TRACE オプションを暗黙の内に指定する。関数
posix_trace_attr_getinherited(),
posix_trace_attr_setinherited()
が存在する。

TRL – _POSIX_TRACE_LOG – _SC_TRACE_LOG **未定義**

このオプションは _POSIX_TRACE オプションを暗黙の内に指定する。関数
posix_trace_attr_getlogfullpolicy(),
posix_trace_attr_getlogsize(),
posix_trace_attr_setlogfullpolicy(),
posix_trace_attr_setlogsize(),
posix_trace_close(),
posix_trace_create_withlog(),
posix_trace_open(),
posix_trace_rewind()
が存在する。

TYM – _POSIX_TYPED_MEMORY_OBJECTS – _SC_TYPED_MEMORY_OBJECT **未定義**

関数
posix_mem_offset(),
posix_typed_mem_get_info(),
posix_typed_mem_open()
が存在する。

—— – _POSIX_VDISABLE **0**

常に存在する (たぶん 0 である)。変更可能な特殊制御文字を設定する値。 これにより特殊制御文字が無効であることを表す。

XOPEN 拡張

_XOPEN_CRYPT, _XOPEN_LEGACY, _XOPEN_REALTIME, _XOPEN_REALTIME_THREADS, _XOPEN_UNIX.

IEEE POSIX1003.1 標準システムコール シグナル関連の規格

古い規格(SVR3)

kill (2) – send a signal to a process or a group of processes
pause (2) – suspend process until signal
signal, sigset, sighold, sigrelse, sigignore, sigpause (2) – simplified signal management

新しい規格(POSIX1)

sigaction (2) – detailed signal management
signaltstack (2) – set or get signal alternate stack context
sigpending (2) – examine signals that are blocked and pending
sigprocmask (2) – change or examine signal mask
sigsend, sigsendset (2) – send a signal to a process or a group of processes
sigsuspend (2) – install a signal mask and suspend process until signal
sigtimedwait, sigwaitinfo (2) – wait for signals
sigwait (2) – wait for a signal to be posted

ANSI の規格

raise (3C) – send signal to program

Linux のリアルタイムシグナル定義

linux の signal.h のリアルタイムシグナル定義は、混乱しているように見えます。

bits/signal.h より

```
#define     _NSIG          65      /* Biggest signal number + 1
                                      (including real-time signals).  */
#define SIGRTMIN        (__libc_current_sigrtmin ())
#define SIGRTMAX        (__libc_current_sigrtmax ())
/* These are the hard limits of the kernel.  These values should not be
   used directly at user level.  */
#define __SIGRTMIN      32
#define __SIGRTMAX      (__NSIG - 1)
```

```
[root@ihawk ~]# kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGSTKFLT
17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
```

```
21) SIGTTIN      22) SIGTTOU      23) SIGURG       24) SIGXCPU
25) SIGXFSZ      26) SIGVTALRM    27) SIGPROF      28) SIGWINCH
29) SIGIO        30) SIGPWR       31) SIGSYS       34) SIGRTMIN
35) SIGRTMIN+1   36) SIGRTMIN+2   37) SIGRTMIN+3   38) SIGRTMIN+4
39) SIGRTMIN+5   40) SIGRTMIN+6   41) SIGRTMIN+7   42) SIGRTMIN+8
43) SIGRTMIN+9   44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12
47) SIGRTMIN+13 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14
51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10
55) SIGRTMAX-9   56) SIGRTMAX-8   57) SIGRTMAX-7   58) SIGRTMAX-6
59) SIGRTMAX-5   60) SIGRTMAX-4   61) SIGRTMAX-3   62) SIGRTMAX-2
63) SIGRTMAX-1   64) SIGRTMAX
```

32 と 33 が未定義です

プログラムで確認すると

```
[root@ihawk ~]# cat t.c
#include <stdio.h>
#include <signal.h>

main()
{
        int     i=0;


        printf("_SIGRTMIN is %d SIGRTMIN is %d¥n",_SIGRTMIN,SIGRTMIN);
        printf("_SIGRTMAX is %d SIGRTMAX is %d¥n",_SIGRTMAX,SIGRTMAX);

        for(i=SIGRTMIN;i<=SIGRTMAX;i++)
        {
                switch(i)
                {
                        case _SIGRTMIN:
                        break;
                        case _SIGRTMAX:
                        break;
                }
        }
}
[root@ihawk ~]# ./t
_SIGRTMIN is 32 SIGRTMIN is 34
_SIGRTMAX is 64 SIGRTMAX is 64
```

また、

```
[root@ihawk ~]# cat t.c
#include <stdio.h>
#include <signal.h>
```

```
main()
{
        int     i=0;


        printf("_SIGRTMIN is %d SIGRTMIN is %d¥n",_SIGRTMIN,SIGRTMIN);
        printf("_SIGRTMAX is %d SIGRTMAX is %d¥n",_SIGRTMAX,SIGRTMAX);

        for(i=SIGRTMIN;i<=SIGRTMAX;i++)
        {
                switch(i)
                {
                        case SIGRTMIN:
                        break;
                        case SIGRTMAX:
                        break;
                }
        }
}
[root@ihawk ~]# make t
cc      t.c   -o t
t.c: In function ‘main’:
t.c:16: error: case ラベルを整数定数に還元できません
t.c:18: error: case ラベルを整数定数に還元できません
make: *** [t] エラー 1
```

理由は、NPTL (Native POSIX Threads Library)

NPTL は内部でリアルタイムシグナル のうち最初の 2 つの番号を使用しており、これらのシグナルはアプリケーションでは使用できない。

したがって、case 文などで、どうしても静的な値を必要とする場合には、SIGRTMAX からの相対値を使用すること！

POSIX シグナル集合の操作

名前

sigemptyset, sigfillset, sigaddset, sigdelset, sigismember – POSIX シグナル集合の操作

書式

```
#include <signal.h>

int sigemptyset(sigset_t *set);
int sigfillset(sigset_t *set);
int sigaddset(sigset_t *set, int signum);
int sigdelset(sigset_t *set, int signum);
int sigismember(const sigset_t *set, int signum);
```

説明

sigsetops(3) 関数群は POSIX シグナル集合(signal set)を操作するため使用する。
sigemptyset() は set で与えられたシグナル集合を空に初期化し、シグナルが 一つも含まれていない状態にする。
sigfillset() は set で与えられたシグナル集合が全てのシグナルを含むようにする。
sigaddset() と sigdelset() は set に signum シグナルをそれぞれ加えたり 、削除したりする。

sigismember() は signum が set に含まれているかどうかをテストする。

返り値

sigemptyset(), sigfillset(), sigaddset(), sigdelset() は成功すれば 0 を 、エラーの場合は -1 を返す。
sigismember() は signum が set に含まれていれば 1 を返し、含まれていなければ 0 を返す。エラーの場合は -1 を返す。

エラー

EINVAL sig が有効なシグナルではない。

pthread のシグナルハンドリング

pthread_sigmask, pthread_kill, sigwait
スレッド内でのシグナルハンドリング

書式

```
#include <pthread.h>
#include <signal.h>
int pthread_sigmask(int how, const sigset_t  *newmask,  sigset_t  *old mask);
int pthread_kill(pthread_t thread, int signo);
int sigwait(const sigset_t *set, int *sig);
```

説明

pthread_sigmask は呼び出しスレッドのシグナルマスクを引数 how および newmask で指定されるように変更する。 oldmask が NULL でないときには、直前のシグナルマスクが oldmask で指し示される領域に格納される。引 数 how および newmask の意味は sigprocmask(2) の引数の意味と同じである。 how が SIG_SETMASK のときには、シグナルマスクが newmask に設定される。 how が SIG_BLOCK のときには、 newmask で指定されるシグナルが現時点のシグナルマスクに追加される。 how が SIG_UNBLOCK のときには、 newmask で指定されるシグナルが現時点のシグナルマスクから取り除かれる。
シグナルマスクはスレッドごとに設定されることを思い出してほしい。しかし sigaction(2) で設定されるシグナルアクションとシグナルハンドラは、すべてのスレッドで共通である。
pthread_kill はシグナル番号 signo のシグナルをスレッド thread に送信する。シグナルは kill(2) に書かれているように配送されハンドルされる。
sigwait は set で指定されるシグナルのうちいずれか 1 つが呼び出しスレ ッド に配送されるまで呼び出しスレッドの実行を停止する。そして受信したシグナルの数を sig で指し示される領域に格納して返る。 set で指定されるシ グナルは sigwait に入るときにブロックされていなければならず、無視されてはならない。配送されたシグナルに対するシグナルハンドラが登録されていてもハンドラ関数は呼び出されない。

PTHREAD 間の同期

- pthread_mutex_lock()/pthread_mutex_unlock()は、mutex 変数 Mutex で、クリティカルセクションを作っています。
- この例でのクリティカルセクションデータは Event で、この値を操作する間、Mutex で保護されています。今、サーバスレッド側で、Evant の値が0である（つまりイベントが無い）とき、スレッドは、pthread_mutex_cond_wait()で、Mutex を解放し、Condition 変数で、眠りにつきます。クライアントスレッド側は、スレッドが Mutex を解放している間に、Event を加算し、pthread_cond_signal()を発行します。
- 休眠状態にあった、スレッドは、クライアントスレッド側が Mutex を解放した時に、pthread_cond_wait()から目覚め、そのとき再び Mutex をロックします。
- この動作で、Mutex 変数 Event はイベントカウンタとして動作するため、イベントを失うことなく、スレッドの実行ー停止を繰り返すことができます。

サーバースレッド

```
pthread_mutex_lock (&Mutex);
if(Event==0)
pthread_cond_wait(&Condition,&Mutex);
Event--;
pthread_mutex_unlock(&Mutex);
```

クライアントスレッド

```
pthread_mutex_lock(&Mutex);
Event++;
pthread_cond_signal(&Condition);
pthread_mutex_unlock(&Mutex);
```

CPU 割り当てとＣＰＵの情報

システムコール
POSIX には規格がない
RedHawk 固有

#include <mpadvise.h>
プロセスの場合
ret = mpadvise (MPA_PRC_GETBIAS,MPA_PID,getpid(),setp);
ret = mpadvise (MPA_PRC_SETBIAS,MPA_PID,getpid(),setp);
ret = mpadvise (MPA_PRC_GETRUN,MPA_PID,getpid(),setp);

pthread の場合
ret = mpadvise (MPA_PRC_GETBIAS,MPA_TID,pthread_self(),setp);
ret = mpadvise (MPA_PRC_SETBIAS,MPA_TID,pthread_self(),setp);
ret = mpadvise (MPA_PRC_GETRUN,MPA_TID,pthread_self(),setp);

Linux 固有（どこで実行しているかを特定出来ない）
名前

sched_setaffinity, sched_getaffinity, CPU_CLR, CPU_ISSET, CPU_SET, CPU_ZERO – プロセスの CPU affinity マスクを設定・取得する

書式

```
#include <sched.h>

int sched_setaffinity(pid_t pid, unsigned int cpusetsize,
                      cpu_set_t *mask);

int sched_getaffinity(pid_t pid, unsigned int cpusetsize,
                      cpu_set_t *mask);

void CPU_CLR(int cpu, cpu_set_t *set);
int CPU_ISSET(int cpu, cpu_set_t *set);
void CPU_SET(int cpu, cpu_set_t *set);
void CPU_ZERO(cpu_set_t *set);
```

説明

プロセスの CPU affinity (親和度) マスクは、そのプロセスが実行を許可されている CPU の集合を決定する。マルチプロセッサ・システムでは、CPU affinity マスクを設定することで性能上のメリットを得られる可能性がある。例えば、特定のプロセスを一つの CPU に括り付け (すなわち、そのプロセスの affinity マスクを一つの CPU に設定し)、他の全てのプロセスの affinity マスクからその CPU を除外することで、確実にそのプロセスの実行速度を最大にすることができる。また、あるプロセスの実行を一つの CPU に限定する ことで、一つの CPU での実行を停止してから別の CPU で実行を再開するときに発生するキャッシュ無効化 (cache invalidation) による性能面の劣化を防ぐこともできる。

CPU affinity マスクは「CPU の集合」を表す cpu_set_t 構造体で表現され、cpu_set_t へのポインタ mask で指定される。CPU 集合を操作するためのマクロが 4 種類提供されている。CPU_ZERO() は集合をクリアする。CPU_SET() と CPU_CLR() はそれぞれ指定された CPU の集合への追加/削除を行う。CPU_ISSET()はある CPU が集合に含まれているかの確認を行う。このマクロは sched_getaffinity() を呼び出した後で使用すると便利である。システム上にある最初の CPU が cpu 値 0 に、次の CPU は cpu 値 1 に 対応し、以後同様である。定数 CPU_SETSIZE (1024) は、CPU 集合に入る CPU 番号の最大値より 1 大きい値を示す。

sched_setaffinity() は、プロセス ID が pid のプロセスの CPU affinity マスクを mask で指定された 値に 設定する。 pid が 0 の場合、呼び出し元プロセスが使われる。引き数 cpusetsize には mask が指すデータの長さ (バイト単位) である。通常は、この引き数には sizeof(cpu_set_t) を指定すればよい。

pid で指定されたプロセスが mask で指定された CPU のいずれかで現在実行されていない場合、そのプロセスは mask で指定された CPU のいずれかに移動される。

sched_getaffinity() は、プロセス ID が pid のプロセスの affinity マスクを mask が指す cpu_set_t 構造体に書き込む。 cpusetsize 引き数には mask の (バイト単位の) 大きさを指定する。

関数 sched_getaffinity() は長さ len のポインタ mask にプロセス pid の affinity マスクを書き込む。 pid が 0 の場合、呼び出し元のプロセスのマスクが返される。

返り値

成功した場合、 sched_setaffinity() と sched_getaffinity() は 0 を返す。エラーの場合は -1 を返し、 errno を適切に設定する。

エラー

EFAULT 指定されたメモリ番地が不正である。

EINVAL affinity ビットマスク mask にシステム上に実際に存在するプロセッサに対応するビットがない 。
またはマスク長 cpusetsize がカーネルで使われている affinity マスクより短い。

EPERM 呼び出し元のプロセスに適切な特権がなかった。 sched_setaffinity() を呼び出すプロセスは、実効ユーザ ID が pid で識別されるプロセスのユーザ ID または実効ユーザ ID と同じであるか 、CAP_SYS_NICE ケーパビリティ (capability) を持たなければならない。

ESRCH プロセス ID pid のプロセスが見つからなかった。

準拠

これらのシステムコールは Linux 固有である。

注意

実際には affinity マスクはスレッド単位の属性で、スレッドグループの各スレッド単位に独立して調整することができる。 gettid(2) コールからの返り値をこのコールの pid 引き数として渡すことができる。

fork(2) 経由で生成された子プロセスは親プロセスの CPU affinity マスクを継承する。 affinity マス クは execve(2) の前後で保存される。

このマニュアルページでは CPU affinity コールの glibc インタフェースを説明している。実際のシステムコール・インタフェースは少し違っており、実際の実装では CPU 集合は簡単なビットマスクであるという実状 を反映し、 mask の型が unsigned long * となっている。成功時には、生の sched_getaffinity() システムコール自身は cpumask_t データ型の (バイト単位の) 大きさを返す。 cpumask_t はカーネル 内 部 で CPU 集合のビットマスクを表現するのに使われているデータ型である。

run コマンドによる割付

作成したアプリケーションは，メインプログラムをシェルで起動し，メインプログラムから，各プロセス（実行モジュール）を起動するようにしている場合に、以下の3行を/etc/rc.local に加えることで、プログラムすることなく、望みのリアルタイム環境で、実行することが出来ます。

①run -b0 -g0
または
run -b0 -a（こちらは、ワーニングが表示されますが問題はありません）
②shield -a1-3
③run -b1-3 -s スケジューリング -P 優先度 プログラム名

説明
①run コマンドのオプション-g0 によって、プロセスグループ0を、-b0 でCPU0に設定する
②shield コマンドオプション-a1-3 によって、CPU1-3 の全ての割り込み(CPU,LOCAL,IRQ)を禁止する
③run コマンドのオプション-b1-3 により、プログラム名(例えば、a.out)を CPU1 から CPU3 のいずれかで実行する。
このとき、-s スケジュール(fifo,rr,other)でスケジューリングを設定し、-P 優先度で優先度を設定する。
また、-s other の場合には、-P は必ず0であり、-s fifo または-s rr の場合には、1 以上 99 以下でなくてはならない。

上記のように、メインプログラムをCPU1～3に割当てた場合，メインプログラムから起動される各プロセス（実行モジュール）は，すべてのプロセスが実行可能状態になった場合に、その瞬間空いている CPU1 から CPU3 のどこかで自動的に実行されます。

もし、実行するプロセス、あるいはスレッドを陽に割り当てたい場合には、プロセス番号か、スレッド番号を指定することで、指定できますが、やや複雑なシェルスクリプトを記述する必要があります。

プロセス番号を指定する場合には、

run -b CPU 番号 -s スケジューリング -P 優先度 -p プロセス番号

であり、

スレッドの場合には、

run -b CPU 番号 -s スケジューリング -P 優先度 -t スレッド番号

です。

システムを含めた各プロセスが，どの CPU に割りついているかを確認するには、やはり run コマンドを利用します。

run コマンドのオプションに-a を付けると、以下のように表示されます。
Bias と Actual は、通常、同じ値ですが、実態を反映しているのは、Actual です。
また、run コマンドで使用するリアルタイム優先度は、Pri の値です。

```
# run -a
Pid   Tid   Bias  Actual  Policy  Pri  Nice  Name
1     1     0xff  0xff    other   0    0     init
2     2     0xff  0xff    other   0    -5    kthreadd
3     3     0x01  0x01    fifo    100  -5    migration/0
4     4     0x01  0x01    other   0    19    krcud/0
5     5     0x01  0x01    fifo    99   -5    ksoftirqd/0
6     6     0x02  0x02    fifo    100  -5    migration/1
7     7     0x02  0x02    other   0    19    krcud/1
8     8     0x02  0x02    fifo    99   -5    ksoftirqd/1
9     9     0x04  0x04    fifo    100  -5    migration/2
10    10    0x04  0x04    other   0    19    krcud/2
11    11    0x04  0x04    fifo    99   -5    ksoftirqd/2
12    12    0x08  0x08    fifo    100  -5    migration/3
13    13    0x08  0x08    other   0    19    krcud/3
14    14    0x08  0x08    fifo    99   -5    ksoftirqd/3
15    15    0x10  0x10    fifo    100  -5    migration/4
16    16    0x10  0x10    other   0    19    krcud/4
17    17    0x10  0x10    fifo    99   -5    ksoftirqd/4
18    18    0x20  0x20    fifo    100  -5    migration/5
19    19    0x20  0x20    other   0    19    krcud/5
20    20    0x20  0x20    fifo    99   -5    ksoftirqd/5
21    21    0x40  0x40    fifo    100  -5    migration/6
22    22    0x40  0x40    other   0    19    krcud/6
23    23    0x40  0x40    fifo    99   -5    ksoftirqd/6
24    24    0x80  0x80    fifo    100  -5    migration/7
25    25    0x80  0x80    other   0    19    krcud/7
26    26    0x80  0x80    fifo    99   -5    ksoftirqd/7
27    27    0x01  0x01    other   0    -5    events/0
28    28    0x02  0x02    other   0    -5    events/1
29    29    0x04  0x04    other   0    -5    events/2
30    30    0x08  0x08    other   0    -5    events/3
31    31    0x10  0x10    other   0    -5    events/4
32    32    0x20  0x20    other   0    -5    events/5
33    33    0x40  0x40    other   0    -5    events/6
34    34    0x80  0x80    other   0    -5    events/7
35    35    0xff  0xff    other   0    -5    khelper
38    38    0xff  0xff    fifo    89   -5    ltmrd
143   143   0x01  0x01    other   0    -5    kblockd/0
144   144   0x02  0x02    other   0    -5    kblockd/1
145   145   0x04  0x04    other   0    -5    kblockd/2
146   146   0x08  0x08    other   0    -5    kblockd/3
147   147   0x10  0x10    other   0    -5    kblockd/4
148   148   0x20  0x20    other   0    -5    kblockd/5
149   149   0x40  0x40    other   0    -5    kblockd/6
150   150   0x80  0x80    other   0    -5    kblockd/7
151   151   0xff  0xff    other   0    -5    kacpid
152   152   0xff  0xff    other   0    -5    kacpi_notify
284   284   0x01  0x01    other   0    -5    ata/0
285   285   0x02  0x02    other   0    -5    ata/1
286   286   0x04  0x04    other   0    -5    ata/2
287   287   0x08  0x08    other   0    -5    ata/3
288   288   0x10  0x10    other   0    -5    ata/4
289   289   0x20  0x20    other   0    -5    ata/5
290   290   0x40  0x40    other   0    -5    ata/6
291   291   0x80  0x80    other   0    -5    ata/7
292   292   0xff  0xff    other   0    -5    ata_aux
293   293   0xff  0xff    other   0    -5    ksuspend_usbd
299   299   0xff  0xff    other   0    -5    khubd
302   302   0xff  0xff    other   0    -5    kseriod
386   386   0xff  0xff    other   0    0     pdflush
387   387   0xff  0xff    other   0    0     pdflush
388   388   0xff  0xff    other   0    -5    kswapd0
389   389   0x01  0x01    other   0    -5    aio/0
390   390   0x02  0x02    other   0    -5    aio/1
391   391   0x04  0x04    other   0    -5    aio/2
392   392   0x08  0x08    other   0    -5    aio/3
393   393   0x10  0x10    other   0    -5    aio/4
394   394   0x20  0x20    other   0    -5    aio/5
395   395   0x40  0x40    other   0    -5    aio/6
396   396   0x80  0x80    other   0    -5    aio/7
399   399   0xff  0xff    other   0    -5    nfsiod
401   401   0x01  0x01    other   0    -5    xfslogd/0
```

```
402   402   0x02  0x02    other   0    -5    xfslogd/1
403   403   0x04  0x04    other   0    -5    xfslogd/2
404   404   0x08  0x08    other   0    -5    xfslogd/3
405   405   0x10  0x10    other   0    -5    xfslogd/4
406   406   0x20  0x20    other   0    -5    xfslogd/5
407   407   0x40  0x40    other   0    -5    xfslogd/6
408   408   0x80  0x80    other   0    -5    xfslogd/7
409   409   0x01  0x01    other   0    -5    xfsdatad/0
410   410   0x02  0x02    other   0    -5    xfsdatad/1
411   411   0x04  0x04    other   0    -5    xfsdatad/2
412   412   0x08  0x08    other   0    -5    xfsdatad/3
413   413   0x10  0x10    other   0    -5    xfsdatad/4
414   414   0x20  0x20    other   0    -5    xfsdatad/5
415   415   0x40  0x40    other   0    -5    xfsdatad/6
416   416   0x80  0x80    other   0    -5    xfsdatad/7
417   417   0xff  0xff    other   0    -5    xfs_mru_cache
418   418   0xff  0xff    other   0    -5    gfs2_scand
419   419   0x01  0x01    other   0    -5    glock_workqueue
420   420   0x02  0x02    other   0    -5    glock_workqueue
421   421   0x04  0x04    other   0    -5    glock_workqueue
422   422   0x08  0x08    other   0    -5    glock_workqueue
423   423   0x10  0x10    other   0    -5    glock_workqueue
424   424   0x20  0x20    other   0    -5    glock_workqueue
425   425   0x40  0x40    other   0    -5    glock_workqueue
426   426   0x80  0x80    other   0    -5    glock_workqueue
1133  1133  0xff  0xff    other   0    -5    netxen
1136  1136  0x01  0x01    other   0    -5    sfc_refill/0
1137  1137  0x02  0x02    other   0    -5    sfc_refill/1
1138  1138  0x04  0x04    other   0    -5    sfc_refill/2
1139  1139  0x08  0x08    other   0    -5    sfc_refill/3
1140  1140  0x10  0x10    other   0    -5    sfc_refill/4
1141  1141  0x20  0x20    other   0    -5    sfc_refill/5
1142  1142  0x40  0x40    other   0    -5    sfc_refill/6
1143  1143  0x80  0x80    other   0    -5    sfc_refill/7
1172  1172  0x01  0x01    other   0    -5    scsi_tgtd/0
1173  1173  0x02  0x02    other   0    -5    scsi_tgtd/1
1174  1174  0x04  0x04    other   0    -5    scsi_tgtd/2
1175  1175  0x08  0x08    other   0    -5    scsi_tgtd/3
1176  1176  0x10  0x10    other   0    -5    scsi_tgtd/4
1177  1177  0x20  0x20    other   0    -5    scsi_tgtd/5
1178  1178  0x40  0x40    other   0    -5    scsi_tgtd/6
1179  1179  0x80  0x80    other   0    -5    scsi_tgtd/7
1192  1192  0xff  0xff    other   0    -5    iscsi_eh
1223  1223  0xff  0xff    other   0    -5    scsi_eh_0
1226  1226  0xff  0xff    other   0    -5    scsi_eh_1
1229  1229  0xff  0xff    other   0    -5    scsi_eh_2
1232  1232  0xff  0xff    other   0    -5    scsi_eh_3
1235  1235  0xff  0xff    other   0    -5    scsi_eh_4
1238  1238  0xff  0xff    other   0    -5    scsi_eh_5
1305  1305  0xff  0xff    other   0    -5    kpsmoused
1319  1319  0x01  0x01    other   0    -5    rpciod/0
1320  1320  0x02  0x02    other   0    -5    rpciod/1
1321  1321  0x04  0x04    other   0    -5    rpciod/2
1322  1322  0x08  0x08    other   0    -5    rpciod/3
1323  1323  0x10  0x10    other   0    -5    rpciod/4
1324  1324  0x20  0x20    other   0    -5    rpciod/5
1325  1325  0x40  0x40    other   0    -5    rpciod/6
1326  1326  0x80  0x80    other   0    -5    rpciod/7
1338  1338  0xff  0xff    other   0    -5    kjournald
1431  1431  0xff  0xff    other   0    -4    udevd
7067  7067  0xff  0xff    other   0    -5    kstriped
7099  7099  0xff  0xff    other   0    -5    kjournald
7100  7100  0xff  0xff    other   0    -5    kjournald
7380  7380  0xff  0xff    other   0    0     syslogd
7383  7383  0xff  0xff    other   0    0     klogd
7396  7396  0xff  0xff    other   0    0     portmap
7417  7417  0xff  0xff    other   0    0     rpc.statd
7431  7431  0xff  0xff    other   0    0     dbus-daemon
7466  7466  0xff  0xff    other   0    0     acpid
7594  7594  0xff  0xff    other   0    0     sshd
7614  7614  0xff  0xff    other   0    0     xinetd
7649  7649  0xff  0xff    other   0    0     crond
7689  7689  0xff  0xff    other   0    0     xfs
```

```
7700  7700  0xff  0xff   other   0    19    anacron
7710  7710  0xff  0xff   other   0    0     atd
12798 12798 0xff  0xff   other   0    0     nshutdownd
12810 12810 0xff  0xff   other   0    0     mingetty
12811 12811 0xff  0xff   other   0    0     mingetty
12812 12812 0xff  0xff   other   0    0     mingetty
12813 12813 0xff  0xff   other   0    0     mingetty
12814 12814 0xff  0xff   other   0    0     mingetty
12815 12815 0xff  0xff   other   0    0     mingetty
12816 12816 0xff  0xff   other   0    0     gdm-binary
13073 13073 0xff  0xff   other   0    0     gdm-binary
13075 13075 0xff  0xff   other   0    0     gdm-rh-security-token-helper
13076 13076 0xff  0xff   other   0    0     Xorg
13256 13256 0xff  0xff   other   0    0     startkde
13293 13293 0xff  0xff   other   0    0     ssh-agent
13326 13326 0xff  0xff   other   0    0     dbus-launch
13327 13327 0xff  0xff   other   0    0     dbus-daemon
13332 13332 0xff  0xff   other   0    0     scim-launcher
13370 13370 0xff  0xff   other   0    0     scim-helper-manager
13371 13371 0xff  0xff   other   0    0     scim-panel-gtk
13371 13372 0xff  0xff   other   0    0     scim-panel-gtk
13373 13373 0xff  0xff   other   0    0     scim-launcher
13383 13383 0xff  0xff   other   0    0     kdeinit
13386 13386 0xff  0xff   other   0    0     dcopserver
13388 13388 0xff  0xff   other   0    0     klauncher
13390 13390 0xff  0xff   other   0    0     kded
13395 13395 0xff  0xff   other   0    0     kwrapper
13397 13397 0xff  0xff   other   0    0     ksmserver
13398 13398 0xff  0xff   other   0    0     kwin
13400 13400 0xff  0xff   other   0    0     kdesktop
13402 13402 0xff  0xff   other   0    0     kicker
13403 13403 0xff  0xff   other   0    0     kio_file
13404 13404 0xff  0xff   other   0    0     kio_media
13406 13406 0xff  0xff   other   0    0     kaccess
13410 13410 0xff  0xff   fifo    50   0     artsd
13415 13415 0xff  0xff   other   0    0     konsole
13416 13416 0xff  0xff   other   0    0     bash
13458 13458 0xff  0xff   other   0    0     knotify
13460 13460 0xff  0xff   other   0    0     run
```

例題集

CPU 割り当てとＣＰＵの情報　例題　affinity.c

```
#include <stdio.h>
#include <sched.h>
#include <errno.h>
#include <string.h>

main()
{
        cpu_set_t mask;
        int ret,cpu;
        unsigned int cpusetsize;
        extern int errno;

        cpusetsize = sizeof(cpu_set_t);
        ret = sched_getaffinity(getpid(),cpusetsize,&mask);
        if (ret!=(-1))
        {
                printf("sched_getaffinity is %08X¥n",mask);
        }
        else
        {
                printf("sched_getaffinity is error %s¥n",strerror(errno));
        }

       for(cpu=0;cpu<8;cpu++)
       {
                _CPU_ZERO(&mask);
                _CPU_SET(cpu,&mask);
                ret = sched_setaffinity(getpid(),cpusetsize,&mask);
                if (ret==0)
                {
                        printf("sched_setaffinity is %08X¥n",mask);
                }
                else
                {
                        printf("sched_setaffinity is error %s¥n",strerror(errno));
                }
       }
}
```

実行結果

```
[root@mss-hawk samples]# ./affinity
sched_getaffinity is 0000000F
sched_setaffinity is 00000001
sched_setaffinity is 00000002
sched_setaffinity is 00000004
sched_setaffinity is 00000008
sched_setaffinity is error Invalid argument
sched_setaffinity is error Invalid argument
sched_setaffinity is error Invalid argument
sched_setaffinity is error Invalid argument
```

CPU 割り当てとＣＰＵの情報　例題　cpuset.c

```
#include <stdio.h>
#include <cpuset.h>

main()
{
        cpuset_t* setp;
        cpu_t max,min;
        char *cpustr;
        int ret,cpu;


        setp = cpuset_alloc();
        cpuset_init(setp);
       if (cpuset_is_empty(setp)) {
            printf("cpu is empty¥n");
        } else {
            printf("cpu is not empty¥n");
        }
        max = cpuset_max_cpu();
        min = cpuset_min_cpu();
        printf("max cpu is %d¥n",(int)(max));
        printf("min cpu is %d¥n",(int)(min));

        for(cpu=min;cpu<=max;cpu++)
        {
                cpuset_set_cpu(setp,cpu,1);/* 1:set */
        }
        cpustr = cpuset_get_string(setp);
        printf("cpu min to max strings is ¥"%s¥"¥n",cpustr);
        free(cpustr);
        ret = cpuset_count(setp);
        printf("cpuset_count %d¥n",ret);
                if (cpuset_is_empty(setp)) {
            printf("cpu is empty¥n");
        } else {
            printf("cpu is not empty¥n");
        }
        cpuset_free(setp);
}
```

実行結果

```
[root@mss-hawk samples]# ./cpuset
cpu is empty
max cpu is 3
min cpu is 0
cpu min to max strings is "f"
cpuset_count 4
cpu is not empty
```

CPU 割り当てとＣＰＵの情報　例題　mpadvise.c

```
#include <stdio.h>
#include <mpadvise.h>
#include <cpuset.h>

main()
{
	cpuset_t* setp;
	cpu_t max,min;
	char *cpustr;
	int ret,cpu;


	setp = cpuset_alloc();
	cpuset_init(setp);
	ret = mpadvise (MPA_CPU_PRESENT,0,0,setp);
	if (ret!=MPA_FAILURE)
	{
		cpustr = cpuset_get_string(setp);
		printf("MPA_CPU_PRESENT is ¥"%s¥"¥n",cpustr);
		free(cpustr);
	}
	else
	{
		printf("MPA_CPU_PRESENT is error¥n");
	}

	cpuset_init(setp);
	ret = mpadvise (MPA_CPU_ACTIVE,0,0,setp);
	if (ret!=MPA_FAILURE)
	{
		cpustr = cpuset_get_string(setp);
		printf("MPA_CPU_ACTIVE is ¥"%s¥"¥n",cpustr);
		free(cpustr);
	}
	else
	{
		printf("MPA_CPU_ACTIVE is error¥n");
	}

	cpuset_init(setp);
	ret = mpadvise (MPA_CPU_BOOT,0,0,setp);
	if (ret!=MPA_FAILURE)
	{
		cpustr = cpuset_get_string(setp);
		printf("MPA_CPU_BOOT is ¥"%s¥"¥n",cpustr);
		free(cpustr);
	}
	else
	{
		printf("MPA_CPU_BOOT is error¥n");
	}
	cpuset_init(setp);
```

```
	ret = mpadvise (MPA_PRC_GETBIAS,MPA_PID,0,setp);
	if (ret!=MPA_FAILURE)
	{
		cpustr = cpuset_get_string(setp);
		printf("MPA_PRC_GETBIAS is ¥"%s¥"¥n",cpustr);
		free(cpustr);
	}
	else
	{
		printf("MPA_PRC_GETBIAS is error¥n");
	}

	cpuset_init(setp);
	ret = mpadvise (MPA_PRC_GETRUN,MPA_PID,0,setp);
	if (ret!=MPA_FAILURE)
	{
		cpustr = cpuset_get_string(setp);
		printf("MPA_PRC_GETRUN is ¥"%s¥"¥n",cpustr);
		free(cpustr);
	}
	else
	{
		printf("MPA_PRC_GETRUN is error¥n");
	}
	max = cpuset_max_cpu();
	min = cpuset_min_cpu();
	for(cpu=min;cpu<=max;cpu++)
	{
		cpuset_init(setp);/* clear */
		cpuset_set_cpu(setp,cpu,1);/* 1:set */
		ret = mpadvise (MPA_PRC_SETBIAS,MPA_PID,0,setp);
		if (ret!=MPA_FAILURE)
		{
			printf("MPA_PRC_SETBIAS %d is SUCCESS¥t",cpu);
		}
		else
		{
			printf("MPA_PRC_SETBIAS %d is FAIL¥t",cpu);
		}
		cpuset_init(setp);
		ret = mpadvise (MPA_PRC_GETRUN,MPA_PID,0,setp);
		if (ret!=MPA_FAILURE)
		{
			cpustr = cpuset_get_string(setp);
			printf("MPA_PRC_GETRUN is ¥"%s¥"¥n",cpustr);
			free(cpustr);
		}
		else
		{
			printf("MPA_PRC_GETRUN is error¥n");
		}
	}
	cpuset_free(setp);
}
```

実行結果

```
[root@mss-hawk samples]# ./mpadvise
MPA_CPU_PRESENT is "f"
MPA_CPU_ACTIVE is "f"
MPA_CPU_BOOT is "1"
MPA_PRC_GETBIAS is "f"
MPA_PRC_GETRUN is "4"
MPA_PRC_SETBIAS 0 is SUCCESS      MPA_PRC_GETRUN is "1"
MPA_PRC_SETBIAS 1 is SUCCESS      MPA_PRC_GETRUN is "2"
MPA_PRC_SETBIAS 2 is SUCCESS      MPA_PRC_GETRUN is "4"
MPA_PRC_SETBIAS 3 is SUCCESS      MPA_PRC_GETRUN is "8"
```

タイマー関連(POSIX1003.1b API)　時間差を計測する例題　　readtime(),adjust()

```
void readtime(struct timespec *nowtime)
{
        clock_gettime(CLOCK_REALTIME,nowtime);
}

void adjust(struct timespec *start,struct timespec *finish, float *realtime)
{
        int     sec,nsec;

        sec = finish->tv_sec - start->tv_sec;
        nsec = finish->tv_nsec - start->tv_nsec;
        if (nsec<0)
        {
                sec --;
                nsec += 1000000000;
        }
        *realtime = (float)(nsec/1000)+(float)(sec*1000000);
}

struct timespec t0,t1;
float usec;
    :
    :
    readtime(&t0);
    ▲
    この部分の時間を計測する
    ▼
    readtime(&t1);
    adjust(&t0,&t1,&usec);
```

インターバルタイマー　と　割り込みハンドラの例題　　timer.c

```
/*
        POSIX RedHawk BenchMark test program version 3.0
*/
#include <stdio.h>
#include <time.h>
#include <signal.h>
#include <errno.h>
#include <sched.h>
#include <sys/types.h>
#include <unistd.h>
#include <mpadvise.h>
#include <cpuset.h>
#include <sys/mman.h>
#include <stdlib.h>
#include <math.h>


#define MAXJITTER (30)
#define BOOTMAXJITTER (100)

extern int errno;
static BOOT_TEST=0;
static FILE *logfd;

//#define HANDLER

void
readtime(struct timespec *nowtime)
{
        clock_gettime(CLOCK_REALTIME,nowtime);
}

void
adjust(struct timespec *start,struct timespec *finish, float *realtime)
{
        int     sec,nsec;

        sec = finish->tv_sec - start->tv_sec;
        nsec = finish->tv_nsec - start->tv_nsec;
        if (nsec<0)
        {
                sec --;
                nsec += 1000000000;
        }
        *realtime = (float)(nsec/1000)+(float)(sec*1000000);
}

timer_t
create_posix_timer(int signum,int sec,int nsec)
{
        static struct sigevent event;
        static timer_t timer;
```

```
        static struct itimerspec itspec;

        /* Create the POSIX timer to generate signum */
        event.sigev_notify = SIGEV_SIGNAL;
        event.sigev_signo = signum;
        if (timer_create((clockid_t)CLOCK_REALTIME,&event,&timer)==(-1))
        {
                fprintf(stderr, "create_posix_timer() Cannot create timer - %s¥n",
                strerror(errno));
                return ((timer_t)-1);
        }
        /*      Set the initial delay and period of the timer.
        This also arms the timer */
         clock_gettime(CLOCK_REALTIME,&itspec.it_value);
        itspec.it_value.tv_sec += 3;
        itspec.it_interval.tv_sec = sec;
        itspec.it_interval.tv_nsec =    nsec;
        if (timer_settime( timer, TIMER_ABSTIME, &itspec, NULL)==(-1))
        {
                return ((timer_t)-1);
        }
        return(timer);
}


#ifdef HANDLER
int loop_flag =1;
void interval_hadler(int no, siginfo_t * info , void * p)
{
        printf("interval_hadler(%d)¥n",no);
}
#endif

#define MSEC 10
#define MAXCOUNT 1001
void timer_test(float jitter)
{
        int signum,status;
        static struct sigaction newact;
        static sigset_t set,oset;
#ifdef HANDLER
        static sigset_t unset;
#endif
        static timer_t timer_id;
         static siginfo_t info;
        static struct timespec t0[MAXCOUNT];
         static float real[MAXCOUNT],min,max,avr;
         static int count = 0;

printf("********************************************** ¥n");
printf("****    posix timer test wait for %d sec **** ¥n",MSEC);
printf("********************************************** ¥n");
        if (sigprocmask(0,NULL,&set)==(-1))
        {
```

```
                fprintf(stderr, "Cannot get sigprocmask - %s¥n",strerror(errno));
                exit(1);
        }
        sigaddset(&set,SIGRTMIN);
        if (sigprocmask(SIG_SETMASK,&set,&oset)==(-1))
        {
                fprintf(stderr, "%s: Cannot set sigprocmask - %s¥n", strerror(errno));
                exit(1);
        }

        signum = SIGRTMIN;
        timer_id = create_posix_timer(signum,0,MSEC * 1000000);
        if (timer_id<0)
        {
                fprintf(stderr, "%s: Cannot set posix timer - %s¥n", strerror(errno));
                exit(1);
         }
         count =0;
#ifdef HANDLER
        sigemptyset(&newact.sa_mask);
        sigaddset(&newact.sa_mask,signum);
        newact.sa_sigaction = interval_hadler;
        newact.sa_flags = SA_SIGINFO|SA_RESTART;
        if (sigaction(signum, &newact, 0)==(-1))
        {
                fprintf(stderr, "%s: Cannot set sigaction - %s¥n", strerror(errno));
                exit(1);
        }
        sigfillset(&unset);
        sigdelset(&unset,SIGRTMIN);
#endif
        do
        {
#ifdef HANDLER
                status = sigsuspend(&unset);
#else
                status = sigwaitinfo(&set,&info);
#endif
         readtime(&t0[count]);
                   count++;
        } while (count<MAXCOUNT);
         timer_delete(timer_id);
         min=1000000000;
         avr =max=0;
        for(count=1;count<MAXCOUNT;count++)
         {
                adjust(&t0[count-1],&t0[count],&real[count]);
                  avr += real[count];
                  if (min> real[count]) min =    real[count];
                  if (max< real[count]) max =    real[count];
                  if (logfd) fprintf(logfd,"%f¥n",real[count]);
         }
         avr /= (float)(MAXCOUNT-1);
         printf(" min %f avr %f max %f¥t",min,avr,max);
```

```
		if (fabs(max-min)<=jitter)
		{
			printf("GOOD(fabs(%f)<%f micro sec)¥n",(max-min),jitter);
		}
		else
		{
			printf("FAIL(fabs(%f)>%f micro sec)¥n",(max-min),jitter);
		}
}

void counter_test(float jitter)
{
		struct timespec t0,t1;
		struct timespec w0={0,0},w1={0,0};
		float real,over;
		int i,usec,nsec,res,ovr,old;


printf("********************************************** ¥n");
printf("****    posix counter test                          **** ¥n");
printf("********************************************** ¥n");
		clock_getres(CLOCK_REALTIME,&w0);
		 res = w0.tv_nsec;
		printf("clock resolution is %d nsec¥n",res);
		 ovr = 0;
		 for(i=0;i<100;i++)
		 {
		 w0.tv_sec=0;
		 w0.tv_nsec= 10*1000*1000;
		 readtime(&t0);
		 nanosleep(&w0,&w1);
		 readtime(&t1);
		 adjust(&t0,&t1,&over);
			ovr += (((int)over * 1000) - w0.tv_nsec);
		 }
		 ovr /= 100;
		printf("nanosleep resolution test overhead is %d nsec¥n",ovr);
		 fflush(stdout);
		for(usec=10;usec<1000000;usec=usec<<1)
		{
			w0.tv_sec=0;
			w0.tv_nsec=1000 * usec - ovr;
			  if (w0.tv_nsec<0) w0.tv_nsec = 0;
			readtime(&t0);
			nanosleep(&w0,&w1);
			readtime(&t1);
			adjust(&t0,&t1,&real);
			printf("request %10d usec real %10.3f usec ret %d nsec¥t",
				usec,real,w1.tv_nsec);
			  if (fabs(real-usec)<=jitter)
			  {
				printf("GOOD(fabs(%f)<%f micro sec)¥n",(real-usec),jitter);
			  }
			  else
```

```
                        {
                                printf("FAIL(fabs(%f)>%f micro sec)¥n",(real-usec),jitter);
                        }
        }
}

int
main(int argc, char **argv)
{
         static struct sched_param myparam;
         cpuset_t* setp;
         int ret;
        extern int errno;
        extern int optind;
        extern char *optarg;
        register int c;
        char    *progname;

        progname = argv[0];
        while((c = getopt(argc, argv, "B1:")) != EOF){
                switch(c) {
                        case 'B':
                                BOOT_TEST=1;
                                break;
                        case 'l':
                                logfd=fopen(optarg,"w");
                                break;
                        default:
                                fprintf(stderr,
                                "usage: %s [-B] B:BOOT_CPU_TEST¥n",
                                progname);
                                exit(1);
                }
        }
#if 0
        if (optind != argc) {
                for (c = optind; c < argc; c++) {
                        if (access(argv[c], 4)) {
                                fprintf(stderr,
                                "%s: ¥"%s¥" %s¥n",
                                  progname,argv[c],strerror(errno));
                                exit(1);
                        }
                }
        }
#endif


         mlockall(MCL_CURRENT|MCL_FUTURE);

         setp = cpuset_alloc();
        myparam.sched_priority = sched_get_priority_max(SCHED_FIFO);
        sched_setscheduler(getpid(),SCHED_FIFO,&myparam);
         system("/usr/bin/shield -r");
```

```
#if 0
		system("/usr/bin/shield -c");
		system("/usr/bin/cpu");
		timer_test(MAXJITTER);
		counter_test(MAXJITTER);
		printf("¥n¥n");
#endif
		system("/usr/bin/shield -p 1 -i 1");
		system("/usr/bin/shield -c");
		system("/usr/bin/cpu");
		cpuset_set_string(setp, "2");
		ret = mpadvise (MPA_PRC_SETBIAS,MPA_PID,0,setp);
		if (ret!=MPA_FAILURE)
	       {
			printf("running 2nd CPU¥n");
		}
		 else
		 {
			printf("Can not running 2nd CPU¥n");
		 }
		 timer_test(MAXJITTER);
		 counter_test(MAXJITTER);
		 printf("¥n¥n¥n");

		 if (BOOT_TEST)
		 {
			cpuset_set_string(setp, "1");
			ret = mpadvise (MPA_PRC_SETBIAS,MPA_PID,0,setp);
			if (ret!=MPA_FAILURE)
		 {
				printf("running boot CPU¥n");
		 }
			else
			{
				printf("Can not running boot CPU¥n");
			}
			timer_test(BOOTMAXJITTER);
			counter_test(BOOTMAXJITTER);
			printf("¥n¥n");
		 }
		 system("/usr/bin/shield -a 1");
		 system("/usr/bin/shield -c");
		 system("/usr/bin/cpu");
		 cpuset_set_string(setp, "2");
		 ret = mpadvise (MPA_PRC_SETBIAS,MPA_PID,0,setp);
		 if (ret!=MPA_FAILURE)
	       {
			printf("running 2nd CPU¥n");
		}
		 else
		 {
			printf("Can not running 2nd CPU¥n");
		 }
		 timer_test(MAXJITTER);
```

```
        counter_test(MAXJITTER);
        if (logfd) fclose(logfd);

}
```

プロセスの優先度（POSIX1003.1b API）　　sched.c

```
#include <stdio.h>
#include <sched.h>
#include <time.h>
#include <sys/types.h>

main()
{
        static struct sched_param myparam;
         int min,max;
         struct timespec rr;


         min = sched_get_priority_min(SCHED_FIFO);
         max = sched_get_priority_max(SCHED_FIFO);
         printf("SCHED_FIFO priority min is %d¥n",min);
         printf("SCHED_FIFO priority max is %d¥n",max);

         min = sched_get_priority_min(SCHED_RR);
         max = sched_get_priority_max(SCHED_RR);
        myparam.sched_priority = min;
        sched_setscheduler(getpid(),SCHED_RR,&myparam);
         sched_rr_get_interval(getpid(),&rr);
         printf("SCHED_RR priority min is %d¥n",min);
         printf("SCHED_RR priority max is %d¥n",max);
         printf("SCHED_RR interval is %d.%d¥n",rr.tv_sec,rr.tv_nsec);

}
```

実行結果

```
[root@RehHawk5 sched]# ./sched
SCHED_FIFO priority min is 1
SCHED_FIFO priority max is 99
SCHED_RR priority min is 1
SCHED_RR priority max is 99
SCHED_RR interval is 0.100006250
```

キュードシグナル(POSIX1003.1b API)　　sigq.c

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/types.h>
#include <time.h>
#include <signal.h>
#include <unistd.h>
#include <fcntl.h>
#define SERVER      1
#define CLIENT      2
#define LOOP        10

volatile sig_atomic_t loop_flag =1;

void display_siginfo(siginfo_t *siginfo)
{
        printf("¥tsi_signo¥t%d¥n¥tsi_code¥t¥t%d¥n¥tsi_pid¥t¥t%d¥n¥tsi_status¥t%d¥n¥tsi_uid¥t¥t%d¥n¥tsi_value¥t%d¥n¥ts
i_addr¥t¥t%d¥n¥tsi_fd¥t¥t%d¥n¥tsi_band¥t¥t%d¥n¥n",
        siginfo->si_signo,
        siginfo->si_code,
        siginfo->si_pid,
        siginfo->si_status,
        siginfo->si_uid,
        siginfo->si_value,
        siginfo->si_addr,
        siginfo->si_fd,
        siginfo->si_band);
        fflush(stdout);
}

void parent_intr_handler(int signo,siginfo_t *siginfo, void *val)
{
        loop_flag ++;
        printf("¥nParent signal handler signo %d val %d¥n",signo,*(int*)val);
        fflush(stdout);
        display_siginfo(siginfo);
}

void server_intr_handler(int signo,siginfo_t *siginfo, void *val)
{
        printf("¥nServer signal handler signo %d val %d¥n",signo,*(int*)val);
        fflush(stdout);
        display_siginfo(siginfo);
}

void client_intr_handler(int signo,siginfo_t *siginfo, void *val)
{
        printf("¥nClient signal handler signo %d val %d¥n",signo,*(int*)val);
        fflush(stdout);
        display_siginfo(siginfo);
}
```

```
void sigint_handler(int signo,siginfo_t *siginfo, void *val)
{
            printf("¥nProcess %d signal handler signo %d val %d¥n",getpid(),signo,(int)val);
            fflush(stdout);
            display_siginfo(siginfo);
           loop_flag = 0;
}

static int server(void)
{
            static struct sigaction newact;
            static struct sigaction intact;
            static union sigval value;
            static sigset_t imask,omask;
            static struct timespec w1 = {1,0}; /* 1 sec */
            static siginfo_t siginfo;
            static int i,err,clientpid=-1;

            sigaddset(&newact.sa_mask,SIGRTMAX);
           newact.sa_sigaction = server_intr_handler;
           newact.sa_flags = SA_SIGINFO|SA_RESTART;
           if (sigaction(SIGRTMAX, &newact, 0)==(-1))
           {
                       fprintf(stderr, "Cannot set sigaction - %s¥n",strerror(errno));
                       exit(1);
           }
            sigaddset(&intact.sa_mask,SIGINT);
           intact.sa_sigaction = sigint_handler;
           intact.sa_flags = SA_SIGINFO|SA_RESTART;
           if (sigaction(SIGINT, &intact, 0)==(-1))
           {
                       fprintf(stderr, "Cannot set sigaction - %s¥n",strerror(errno));
                       exit(1);
           }
            sigemptyset(&imask);
            sigaddset(&imask,SIGINT);
            sigaddset(&imask,SIGRTMAX);
           if (sigprocmask(SIG_BLOCK,&imask,&omask)==(-1))
           {
                       fprintf(stderr, "Cannot set sigprocmask - %s¥n",strerror(errno));
                       exit(1);
           }
            nanosleep(&w1,NULL);
            value.sival_int = getpid();
            printf("sigqueue from Server Process %d to Parent Process %d value %d¥n",
            getpid(),getppid(), value.sival_int);
            fflush(stdout);
            sigqueue(getppid(),SIGRTMAX,value);
            while(loop_flag==1)
            {
                        err = sigwaitinfo(&imask,&siginfo);
                        printf("¥nServer sigwaitinfo return code %d errno %d ¥"%s¥"¥n",err,errno,strerror(errno));
                        display_siginfo(&siginfo);
                        switch (siginfo.si_signo)
```

```
                {
                        case SIGINT: loop_flag=0;break;
                        case _SIGRTMAX:
                                if (clientpid==(-1)&&(siginfo.si_pid==getppid()))
                                {
                                        clientpid = siginfo.si_int;
                                        loop_flag = 2;
                                }
                        break;
                }
        }
        sigfillset(&imask);
        sigdelset(&imask,SIGINT);
        sigdelset(&imask,SIGRTMAX);
        for(i=0;i<LOOP;i++)
        {
                sigsuspend(&imask);
        }
        for(i=0;i<LOOP;i++)
        {
                value.sival_int = i;
                sigqueue(clientpid,SIGRTMAX,value);
        }
        printf("ServerClient Process exit¥n");
        fflush(stdout);
        return(0);
}

static int client(void)
{
        static struct sigaction newact;
        static struct sigaction intact;
        static sigset_t imask,omask;
        static union sigval value;
        static struct timespec w1 = {1,0}; /* 1 sec */
        static siginfo_t siginfo;
        static int i,err,serverpid=-1;

        sigaddset(&newact.sa_mask,SIGRTMAX);
       newact.sa_sigaction = client_intr_handler;
       newact.sa_flags = SA_SIGINFO|SA_RESTART;
       if (sigaction(SIGRTMAX, &newact, 0)==(-1))
       {
                fprintf(stderr, "Cannot set sigaction - %s¥n",strerror(errno));
                exit(1);
       }
        sigaddset(&intact.sa_mask,SIGINT);
       intact.sa_sigaction = sigint_handler;
       intact.sa_flags = SA_SIGINFO|SA_RESTART;
       if (sigaction(SIGINT, &intact, 0)==(-1))
       {
                fprintf(stderr, "Cannot set sigaction - %s¥n",strerror(errno));
                exit(1);
       }
```

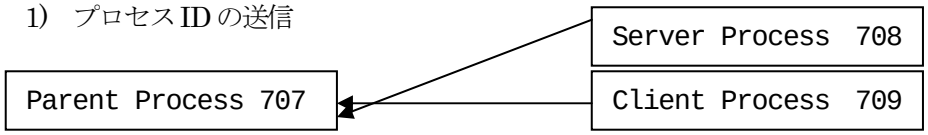
```
	sigemptyset(&imask);
	sigaddset(&imask,SIGINT);
	sigaddset(&imask,SIGRTMAX);
	if (sigprocmask(SIG_BLOCK,&imask,&omask)==(-1))
	{
		fprintf(stderr, "Cannot set sigprocmask - %s¥n",strerror(errno));
		exit(1);
	}
	nanosleep(&w1,NULL);
	value.sival_int = getpid();
	printf("sigqueue from Client Process %d to Parent Process %d value %d¥n",
	getpid(),getppid(), value.sival_int);
	fflush(stdout);
	sigqueue(getppid(),SIGRTMAX,value);

	while(loop_flag==1)
	{
		err = sigwaitinfo(&imask,&siginfo);
		printf("¥nClient sigwaitinfo return code %d errno %d ¥"%s¥"¥n",err,errno,strerror(errno));
		display_siginfo(&siginfo);
		switch (siginfo.si_signo)
		{
			case SIGINT: loop_flag=0;break;
			case _SIGRTMAX:
				if (serverpid==(-1)&&(siginfo.si_pid==getppid()))
				{
					serverpid = (int)siginfo.si_int;
					loop_flag = 2;
				}
			break;
		}
	}
	sigfillset(&imask);
	sigdelset(&imask,SIGINT);
	sigdelset(&imask,SIGRTMAX);
	for(i=0;i<LOOP;i++)
	{
		printf("send from client data %d¥n",i);
		printf("¥n**********¥nany input is start");fflush(stdout); getchar();
		value.sival_int = i;
		sigqueue(serverpid,SIGRTMAX,value);
	}
	for(i=0;i<LOOP;i++)
	{
		sigsuspend(&imask);
	}
	printf("Client Process exit¥n");
	fflush(stdout);
	return(0);
}

int do_main(int option)
{
	static struct sigaction newact;
```

```
	static struct sigaction intact;
	pid_t ServerPID;
	pid_t ClientPID;
	int status;
	static union sigval value;
	static sigset_t imask,omask;
	static struct timespec w1 = {1,0}; /* 1 sec */

	sigaddset(&newact.sa_mask,SIGRTMAX);
	newact.sa_sigaction = parent_intr_handler;
	newact.sa_flags = SA_SIGINFO|SA_RESTART;
	if (sigaction(SIGRTMAX, &newact, 0)==(-1))
	{
		fprintf(stderr, "Cannot set sigaction - %s¥n",strerror(errno));
		return(1);
	}

	printf("Parent Process ID %d¥n",getpid());fflush(stdout);

	sigaddset(&intact.sa_mask,SIGINT);
	intact.sa_sigaction = sigint_handler;
	intact.sa_flags = SA_SIGINFO|SA_RESTART;
	if (sigaction(SIGINT, &intact, 0)==(-1))
	{
		fprintf(stderr, "Cannot set sigaction - %s¥n",strerror(errno));
		return(1);
	}
	sigemptyset(&imask);
	sigaddset(&imask,SIGINT);
#ifndef TEST1
	if (sigprocmask(SIG_BLOCK,&imask,&omask)==(-1))
	{
		fprintf(stderr, "Cannot set sigprocmask - %s¥n",strerror(errno));
		return(1);
	}
#endif
	ServerPID=fork();
	if (ServerPID<0)
	{
		fprintf(stderr,"server fork error¥n");
		return(0);
	}
	if (ServerPID==0)
	{
		server();
		exit(0);
	}
	printf("Server Process ID %d¥n",ServerPID);fflush(stdout);
	ClientPID=fork();
	if (ClientPID<0)
	{
		fprintf(stderr,"client fork error¥n");
		return(0);
	}
```

```
        if (ClientPID==0)
        {
                client();
                exit(0);
        }
        printf("Client Process ID %d¥n",ClientPID);fflush(stdout);

        while(loop_flag!=3)
        {
                sigsuspend(&imask);
        }
        printf("sigqueue from Parent Process %d to Server Process %d value %d¥n",getpid(),ServerPID,ClientPID);
        fflush(stdout);
        printf("¥n*********************¥nany input is start");fflush(stdout); getchar();
        value.sival_int = ClientPID;
        sigqueue(ServerPID,SIGRTMAX,value);
        nanosleep(&w1,NULL);

        printf("sigqueue from Parent Process %d to Client Process %d value %d¥n",getpid(),ClientPID,ServerPID);
        fflush(stdout);
        value.sival_int = ServerPID;
        sigqueue(ClientPID,SIGRTMAX,value);

        waitpid(ClientPID,&status,0);
        waitpid(ServerPID,&status,0);
        printf("Parent Process exit!¥n");
        return(0);
}

main( int argc, char **argv )
{
        extern int optind;
        extern char *optarg;
        register int c, option = 0;

        while((c = getopt(argc, argv, "sc")) != EOF)
        {
                switch(c)
                {
                        case 'c':
                                option|=CLIENT;
                                break;
                        case 's':
                                option|=SERVER;
                                break;
                        default:
                                break;
                }
        }
        return(do_main(option));
}
```

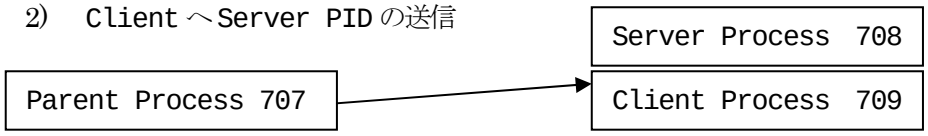
実行結果

1) プロセスIDの送信



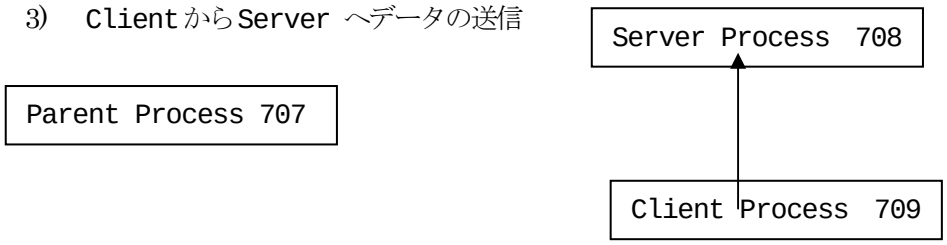

```
Parent Process ID 707
Server Process ID 708
Client Process ID 709
sigqueue from Server Process 708 to Parent Process 707 value 708

Parent signal handler signo 52 val 15
       si_signo       52
       si_code        -1
       si_pid         708
       si_status      708
       si_uid         0
       si_value       708
       si_addr        708
       si_fd          708
       si_band        0

sigqueue from Client Process 709 to Parent Process 707 value 709

Parent signal handler signo 52 val 15
       si_signo       52
       si_code        -1
       si_pid         709
       si_status      709
       si_uid         0
       si_value       709
       si_addr        709
       si_fd          709
       si_band        0

sigqueue from Parent Process 707 to Server Process 708 value 709
```

2) Client へ Server PID の送信




```
Server sigwaitinfo return code 52 errno 6 "No such device or address"
       si_signo       52
       si_code        -1
       si_pid         707
       si_status      709
       si_uid         0
       si_value       709
       si_addr        707
       si_fd          707
       si_band        0

sigqueue from Parent Process 707 to Client Process 709 value 708

Client sigwaitinfo return code 52 errno 6 "No such device or address"
       si_signo       52
       si_code        -1
       si_pid         707
       si_status      708
       si_uid         0
       si_value       708
       si_addr        707
       si_fd          707
       si_band        0
```

3) ClientからServer へデータの送信




```
send from client data 0
Server signal handler signo 52 val 15
       si_signo       52
       si_code        -1
       si_pid         709
       si_status       0
       si_uid         0
       si_value        0
       si_addr        709
       si_fd          709
       si_band        0

send from client data 2
Server signal handler signo 52 val 15
       si_signo       52
       si_code        -1
       si_pid         709
       si_status       1
       si_uid         0
       si_value        1
       si_addr        709
       si_fd          709
       si_band        0
 :
 :
Client signal handler signo 52 val 15
       si_signo       52
       si_code        -1
       si_pid         708
       si_status       9
       si_uid         0
       si_value        9
       si_addr        708
       si_fd          708
       si_band        0

Client Process exit
Parent Process exit
```

シェアードメモリ(POSIX1003.1b API)　　shm_open.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/param.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <fcntl.h>
#include <errno.h>

extern int errno;

int
main(int argc, char *argv[])
{

	int		oflg;
	int		memfd;
	mode_t		omode;
	const char *shmname = "/shared_memory";
	struct	shared_memory
	{
		int	a, b, counter, initial;
	};
	struct	shared_memory *shmaddr = NULL;


	/* set paramater */
	oflg	= O_CREAT | O_RDWR| O_TRUNC;
	omode = 0644;

	/* open shared memory */
	memfd = shm_open(shmname, oflg, omode );			/* /dev/shm/shared_memory が作られる */
	if(memfd==-1){
		printf("Error: shm_open() Error!! [%d]¥n",errno);
		perror(NULL);
		return (-1);
	}

	printf("shm_open() OK!! memfd[%d]¥n",memfd);
	/* set the memory object's size */
	if(ftruncate(memfd,sizeof(struct shared_memory))==-1)
	{
	   fprintf(stderr,"ftruncate : %s¥n",strerror(errno));
	   return EXIT_FAILURE;
	}

	/* map the memory object */
	shmaddr=mmap(0,sizeof(struct	shared_memory),PROT_READ | PROT_WRITE,MAP_SHARED,memfd,0);
	if((void *)shmaddr==MAP_FAILED)
	{
	   fprintf(stderr,"mmap faild : %s¥n",strerror(errno));
	   return EXIT_FAILURE;
	}
```

```
    printf("map addr is %#x¥n",shmaddr);
    printf("input any keys continue:");fflush(stdout);
    getchar();
    printf("OK¥n");
    if (munmap(shmaddr,sizeof(struct    shared_memory))<0)
    {
        fprintf(stderr,"munmap faild : %s¥n",strerror(errno));
    }
    if (close(memfd)<0)
    {
        fprintf(stderr,"close faild : %s¥n",strerror(errno));
    }
    shm_unlink(shmname);

    return (0);
}
```

メッセージキュー(POSIX1003.1b API)　　mq.c

```
#include <stdio.h>
#include <limits.h>
#include <mqueue.h>
#include <signal.h>
#include <stdlib.h>
#include <errno.h>

static mqd_t fbq;

void
receive_mq(int signo,siginfo_t *siginfo, void *val)
{
        unsigned int recv,msg_prio;
        struct sigevent notification;


printf("¥tsi_signo¥t%d¥n¥tsi_code¥t¥t%d¥n¥tsi_pid¥t¥t%d¥n¥tsi_status¥t%d¥n¥tsi_uid¥t¥t%d¥n¥tsi_value¥t%d¥n¥tsi_addr¥t¥t%
d¥n¥tsi_fd¥t¥t%d¥n¥tsi_band¥t¥t%d¥n¥n",
        siginfo->si_signo,
        siginfo->si_code,
        siginfo->si_pid,
        siginfo->si_status,
        siginfo->si_uid,
        siginfo->si_value,
        siginfo->si_addr,
        siginfo->si_fd,
        siginfo->si_band);
        fflush(stdout);

        notification.sigev_notify = SIGEV_SIGNAL;
        notification.sigev_signo = SIGRTMIN;
        mq_notify(fbq,&notification);

                mq_receive(fbq,(char*)&recv,sizeof(int),&msg_prio);
                printf("data %d priority %d¥n",recv,msg_prio);fflush(stdout);
}

main()
{
        struct mq_attr comattr;
        unsigned int i,send,recv,msg_prio;
        struct sigaction newact;
        struct sigevent notification;
        int signum;
        static sigset_t set,oset,eiset;


        /* メッセージキュー  fbq */
        comattr.mq_flags = O_NONBLOCK;//注意  このフラグは設定しても無視される
        comattr.mq_maxmsg = 100;
        comattr.mq_msgsize = sizeof(int);
        comattr.mq_curmsgs = 0;
```

```
printf("Message Queue Polling Test¥n");
        if ((fbq = mq_open("/fbq",O_RDWR|O_CREAT|O_NONBLOCK,0666, &comattr))==(mqd_t)(-1))
        {//open 時の O_NONBLOCK だけが有効
                fprintf(stderr,"%s¥n",strerror(errno));
              exit(1);
        }

         printf("send message queue      :");
         for (i = 0; i<10;i++)
         {
                send = i;
                if(mq_send(fbq,(char*)&send,sizeof(int),i)<0)
                {
                        fprintf(stderr,"%s¥n",strerror(errno));
                }
                printf("%d ",i);fflush(stdout);
         }
         printf("¥n");

         printf("receive message queue :");
         for (i = 0; i<10;i++)
         {
                mq_receive(fbq,(char*)&recv,sizeof(int),&msg_prio);
                printf("%x:%x ",msg_prio,recv);fflush(stdout);
         }
         printf("¥n");

printf("Message Queue Interrupt Test¥n");
        signum = SIGRTMIN;
        sigaddset(&newact.sa_mask,signum);
        newact.sa_sigaction = receive_mq;
        newact.sa_flags = SA_SIGINFO|SA_RESTART;
        sigaction(signum, &newact, 0);

        notification.sigev_notify = SIGEV_SIGNAL;
        notification.sigev_signo = signum;
         mq_notify(fbq,&notification);

printf("send 10¥n");
         send = 10;
         mq_send(fbq,(char*)&send,sizeof(int),1);
         sleep(1);
printf("send 9¥n");
         send = 9;
         mq_send(fbq,(char*)&send,sizeof(int),1);
         sleep(1);
         mq_close(fbq);
        mq_unlink("/fbq");

}
```

実行結果

```
[root@RehHawk5 mq]# ./mq
Message Queue Polling Test
send message queue       :0 1 2 3 4 5 6 7 8 9
receive message queue :9:9 8:8 7:7 6:6 5:5 4:4 3:3 2:2 1:1 0:0
Message Queue Interrupt Test
send 10
        si_signo        34
        si_code          -3
        si_pid          16048
        si_status       2100
        si_uid          0
        si_value        2100
        si_addr          16048
        si_fd           0
        si_band          16048

data 10 priority 1
send 9
        si_signo        34
        si_code          -3
        si_pid          16048
        si_status       1
        si_uid          0
        si_value        1
        si_addr          16048
        si_fd           0
        si_band          16048

data 9 priority 1
```

注意

/proc インタフェース

以下のインタフェースを使って、POSIX メッセージキューが消費するカーネルメモリの量を制限することができる。

/proc/sys/fs/mqueue/msg_max

こ のファイルを使って、一つのキューに入れられるメッセージの最大数の上限値を参照したり変更したりできる。この値は、 mq_open(3) に渡す attr->mq_maxmsg 引き数に対する上限値として機能する。 msg_max のデフォルト値および最小値は 10 である; 上限は「埋め込みの固定値」(HARD_MAX) で (131072 / sizeof(void *)) (Linux/86 では 32768) である。この上限 は特 権プロセス (CAP_SYS_RESOURCE) では無視されるが、埋め込みの固定値による上限はどんな場合にでも適用される。

/proc/sys/fs/mqueue/msgsize_max

このファイルを使って、メッセージの最大サイズの上限値を参照したり変更し た り で き る。
この値は、mq_open(3)に渡す attr.mq_msgsize 引き数に対する上限値として機能する。msgsize_max のデフォルト値および 最小値は 8192 バイトである; この上限は特権プロセス(CAP_SYS_RESOURCE)では無視される。

/proc/sys/fs/mqueue/queues_max

こ のファイルを使って、作成することができるメッセージキューの数に対するシステム全体での制限を参照したり変更したりできる。一度この上限に達すると、新しいメッセージキュー を作成できるのは特権プロセス(CAP_SYS_RESOURCE)だけとなる。queues_max のデフォルト値は 256 であり、0 から INT_MAX の範囲の任意の値に変更することができる。

64 ビットリアルタイムファイル(POSIX1003.1b API)　　rfile.c

```
#define _LARGEFILE64_SOURCE
#include <stdio.h>
#define _USE_GNU
#include <sys/types.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
```

この順序が必須

```
main()
{
        int fd;
        off64_t align;
        long long bytes;
        char *dummy;

        bytes = 4294967296LL;//0x100000000;
        if ((fd = open64("realtime_file",(O_CREAT|O_WRONLY|O_DIRECT), 0666)) <0)        {
                exit(1);
        }
        align = (off64_t)fpathconf(fd,_PC_ALLOC_SIZE_MIN);
        printf("_PC_ALLOC_SIZE_MIN %lld¥n",align);
        printf("allocate %lld bytes¥n",bytes);
        posix_memalign((void*)&dummy,align,0x1000);
        if (posix_fadvise(fd,0LL,bytes,POSIX_FADV_NOREUSE)<0)        {
                fprintf(stderr,"fadvise error¥n");
                exit(0);
        } else {
                if (posix_fallocate64(fd,0LL,bytes)<0)
                {
                        fprintf(stderr, "falocate error¥n");
                        exit(0);
                }
                bytes -= 4096LL;
                if (lseek64(fd,bytes,SEEK_SET)<0)
                {
                        fprintf(stderr,"lseek error¥n");
                }
                write(fd,dummy,0x1000);
        }
        close(fd);
}
```

実行結果

```
[root@RehHawk5 contig]# ./rfile
_PC_ALLOC_SIZE_MIN 4096
allocate 4294967296 bytes
[root@RehHawk5 contig]# ls -l
合計 440
-rw-r--r-- 1 root sys              408  8 月  17 17:30 Makefile
-rw-r--r-- 1 root root 4294967296  8 月  18 09:06 realtime_file
-rwxr-xr-x 1 root root            6211  8 月  18 09:06 rfile
-rw-r--r-- 1 root sys              897  8 月  18 09:06 rfile.c
-rw-r--r-- 1 root root            1800  8 月  18 09:06 rfile.o
```

非同期ファイル I/O(POSIX1003.1b API)

```
#include <stdio.h>
#include <stdlib.h>
#define _USE_GNU
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <aio.h>
#include <sys/mman.h>

#define DATA_BUF_SIZE 4096
#define DATA_BUF_NUM 128

aio_test(int fd)
{
 int n,status;
 unsigned char *   Aio_buff[DATA_BUF_NUM];
 struct aiocb Aiocb[DATA_BUF_NUM];
 struct aiocb *List[DATA_BUF_NUM];
 struct timespec timeout = {10,0};
        struct sigevent sig;

        for(n=0;n<DATA_BUF_NUM;n++)
        {
                Aio_buff[n] = (unsigned char*)memalign(sysconf(_SC_PAGESIZE),DATA_BUF_SIZE);
                    memset((void *)(Aio_buff[n]),n,DATA_BUF_SIZE);


                Aiocb[n].aio_buf = Aio_buff[n];
           Aiocb[n].aio_offset = (long long)(DATA_BUF_SIZE * n);
                Aiocb[n].aio_nbytes          = (long long)DATA_BUF_SIZE;
           Aiocb[n].aio_fildes       = fd;
                Aiocb[n].aio_reqprio        = 0;
                Aiocb[n].aio_lio_opcode   = LIO_WRITE;
                Aiocb[n].aio_sigevent.sigev_notify = SIGEV_NONE;
           List[n] = &Aiocb[n];

 }
 mlockall(MCL_CURRENT|MCL_FUTURE);

 lio_listio(LIO_WAIT,(struct aiocb **)&List[0],DATA_BUF_NUM, &sig);
 aio_suspend((const struct aiocb **)&List[0],DATA_BUF_NUM, &timeout);
 munlockall();

        for(n=0;n<DATA_BUF_NUM;n++)
 {
           status = aio_return(&(Aiocb[n]));
           if (status!=DATA_BUF_SIZE) printf("%d is write error¥n",n);
 }
        for(n=0;n<DATA_BUF_NUM;n++)
 {
           status = aio_error(&(Aiocb[n]));
           if (status) printf("%d is error %d:%s¥n",n,status,strerror(status));
 }

}

main()
{
 int fd;
 long long bytes;
 int dummy=0;
```

```
bytes = DATA_BUF_SIZE * DATA_BUF_NUM ;

if ((fd = open("realtime_file",(O_CREAT|O_DIRECT|O_WRONLY|O_SYNC|O_DSYNC|O_RSYNC), 0666)) <0)
{
        exit(1);
}

printf("allocate %lld bytes¥n",bytes);

if (posix_fadvise(fd,0,bytes,POSIX_FADV_NOREUSE)<0)
{
        fprintf(stderr,"fadvise error¥n");
        exit(0);
} else {
        if (posix_fallocate(fd,0,bytes)<0)
        {
                fprintf(stderr, "falocate error¥n");
                exit(0);
        }
        bytes -= (long long)sizeof(int);
        lseek(fd,bytes,SEEK_SET);
        write(fd,&dummy,4);
        lseek(fd,0,SEEK_SET);
}
aio_test(fd);
close(fd);
}
```

実行結果

```
[root@RehHawk5 aio]# ./aio
allocate 524288 bytes

 [root@RehHawk5 aio]# ll
合計 536
-rw-r--r-- 1 root sys       450  8 月  18 04:51 Makefile
-rwxr-xr-x 1 root root     7484  8 月  18 04:52 aio
-rw-r--r-- 1 root sys      2162  8 月  18 04:51 aio.c
-rw-r--r-- 1 root root     2452  8 月  18 04:52 aio.o
-rw-r--r-- 1 root root 524288  8 月  18 04:52 realtime_file
```

実行中のプロセスステータスのスナップショット

```
F S UID        PID  PPID  LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
4 S root      5233  4614  5233  0    1  80   0 -  1515 wait   02:00 pts/2    00:00:00 bash
4 S root     14886  5233 14886  0    2  80   0 -   682 futex_ 04:52 pts/2    00:00:00 ./aio
1 D root     14886  5233 14888  0    2  80   0 -   682 blockd 04:52 pts/2    00:00:00 ./aio
0 R root     14887  5233 14887  0    1  80   0 -  1290 -      04:52 pts/2    00:00:00 ps -flL
```

セマフォ(POSIX1003.1b API)　　sem.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <semaphore.h>
#include <errno.h>

main()
{
 sem_t *sem;
 int pshared=2;          /* 2 process shared */
 int value=0;
 int sval,status;

 sem = sem_open("/posix_sem0",O_CREAT|O_EXCL,0644,0); //create /dev/shm/sem.posix_sem0
 if (sem==(sem_t*)(-1))
 {
        printf("error %s¥n",strerror(errno));
        exit(0);
 }
 sem_init(sem,pshared,value);

 sem_getvalue(sem,&sval); printf("initial semaphore value is %d¥n",sval);

 sem_post(sem);
 sem_getvalue(sem,&sval); printf("after sem_post semaphore value is %d¥n",sval);

 sem_wait(sem);
 sem_getvalue(sem,&sval); printf("after sem_wait semaphore value is %d¥n",sval);

 status = sem_trywait(sem);
 if (status)
        printf("status %d %s¥n",status,strerror(errno));
 else
        printf("successful lock a counting semaphore¥n");
 sem_getvalue(sem,&sval); printf("current semaphore value is %d¥n",sval);

 sem_post(sem);
 status = sem_trywait(sem);
 if (status)
        printf("status %d %s¥n",status,strerror(errno));
 else
        printf("successful lock a counting semaphore¥n");
 sem_getvalue(sem,&sval); printf("current semaphore value is %d¥n",sval);

 sem_close(sem);
 sem_unlink("/posix_sem0");
}
```

実行結果

```
[root@RehHawk5 sem]# ./sem
initial semaphore value is 0
after sem_post semaphore value is 1
after sem_wait semaphore value is 0
status -1 Resource temporarily unavailable
current semaphore value is 0
successful lock a counting semaphore
current semaphore value is 0
```

pthread によるミニスケジューラ(POSIX1003.1c API)　　tswitch.c

```
#include <stdio.h>
#include <errno.h>
#include <sys/types.h>
#include <time.h>
#include <signal.h>
#include <string.h>
#include <unistd.h>
#include <mpadvise.h>
#include <pthread.h>
#include <locale.h>
#include <cpuset.h>
#include <stdlib.h>
#include <math.h>

#define CYCLE                   1000
#define THREAD_NO       3

char   *progname,*lang;

volatile sig_atomic_t loop_flag=1,GlobalCounter=0;
sigset_t diset,oset,eiset;

#define TESTNUMBER      10000
#define MAXJITTER       (30)

pthread_cond_t Condition[THREAD_NO];
#define COND_SIGNAL     pthread_cond_signal
#define COND_WAIT       pthread_cond_wait
pthread_mutex_t Mutex[THREAD_NO];
#define MUTEX_LOCK      pthread_mutex_lock
#define MUTEX_UNLOCK    pthread_mutex_unlock

int Event[THREAD_NO]={0,0,0};
float SwitchTime1[TESTNUMBER/2];
float SwitchTime2[TESTNUMBER/2];
float IntervalTime[TESTNUMBER];
struct timespec StartTime={0,0},EndTime={0,0};
struct timespec CurrrentTime={0,0};

void
readtime(struct timespec *nowtime)
{
        clock_gettime(CLOCK_REALTIME,nowtime);
}

void
adjust(struct timespec *start,struct timespec *finish, float *realtime)
{
        int         sec,nsec;

        sec = finish->tv_sec - start->tv_sec;
        nsec = finish->tv_nsec - start->tv_nsec;
        if (nsec<0)
        {
                        sec --;
                        nsec += 1000000000;
        }
        *realtime = (float)(nsec/1000)+(float)(sec*1000000);
}


timer_t
```

```
create_posix_timer(int signum,int sec,int nsec)
{
        static struct sigevent event;
    static timer_t timer;
        static struct itimerspec itspec;

    /* Create the POSIX timer to generate signum */
        event.sigev_notify = SIGEV_SIGNAL;
        //event.sigev_notify = SIGEV_THREAD;
        event.sigev_signo = signum;
    if (timer_create((clockid_t)CLOCK_REALTIME,&event,&timer)==(-1))
        {
                fprintf(stderr, "create_posix_timer() Cannot create timer - %s¥n",
                strerror(errno));
            return ((timer_t)-1);
        }
    /*      Set the initial delay and period of the timer.
        This also arms the timer */
    clock_gettime(CLOCK_REALTIME,&itspec.it_value);
        itspec.it_value.tv_sec += 1;
        itspec.it_interval.tv_sec = sec;
        itspec.it_interval.tv_nsec =    nsec;
        if (timer_settime( timer, TIMER_ABSTIME, &itspec, NULL)==(-1))
        {
                return ((timer_t)-1);
        }
    return(timer);
}

void interval_hadler(int sig)
{
#ifdef DEBUG
        printf("interval_hadler(%d)¥n",GlobalCounter);
#endif
    if(Event[0]!=THREAD_NO) return;

    readtime(&StartTime);
    if(CurrrentTime.tv_sec)
    {
                adjust(&CurrrentTime,&StartTime,&IntervalTime[GlobalCounter]);
    }
    CurrrentTime.tv_sec = StartTime.tv_sec;
    CurrrentTime.tv_nsec = StartTime.tv_nsec;
    switch(GlobalCounter%10)
    {
                case 0:
                        MUTEX_LOCK(&Mutex[1]); /* lock */
                                Event[1]++;
                                COND_SIGNAL(&Condition[1]);
                        MUTEX_UNLOCK(&Mutex[1]); /* unlock */
                break;
                case 1:
                        MUTEX_LOCK(&Mutex[2]); /* lock */
                                Event[2]++;
                                COND_SIGNAL(&Condition[2]);
                        MUTEX_UNLOCK(&Mutex[2]); /* unlock */
                break;
                case 2:
                        MUTEX_LOCK(&Mutex[1]); /* lock */
                                Event[1]++;
                                COND_SIGNAL(&Condition[1]);
                        MUTEX_UNLOCK(&Mutex[1]); /* unlock */
                break;
                case 3:
                        MUTEX_LOCK(&Mutex[2]); /* lock */
```

```
                                        Event[2]++;
                                        COND_SIGNAL(&Condition[2]);
                                MUTEX_UNLOCK(&Mutex[2]); /* unlock */
                        break;
                        case 4:
                                MUTEX_LOCK(&Mutex[1]); /* lock */
                                        Event[1]++;
                                        COND_SIGNAL(&Condition[1]);
                                MUTEX_UNLOCK(&Mutex[1]); /* unlock */
                        break;
                        case 5:
                                MUTEX_LOCK(&Mutex[2]); /* lock */
                                        Event[2]++;
                                        COND_SIGNAL(&Condition[2]);
                                MUTEX_UNLOCK(&Mutex[2]); /* unlock */
                        break;
                        case 6:
                                MUTEX_LOCK(&Mutex[1]); /* lock */
                                        Event[1]++;
                                        COND_SIGNAL(&Condition[1]);
                                MUTEX_UNLOCK(&Mutex[1]); /* unlock */
                        break;
                        case 7:
                                MUTEX_LOCK(&Mutex[2]); /* lock */
                                        Event[2]++;
                                        COND_SIGNAL(&Condition[2]);
                                MUTEX_UNLOCK(&Mutex[2]); /* unlock */
                        break;
                        case 8:
                                MUTEX_LOCK(&Mutex[1]); /* lock */
                                        Event[1]++;
                                        COND_SIGNAL(&Condition[1]);
                                MUTEX_UNLOCK(&Mutex[1]); /* unlock */
                        break;
                        case 9:
                                MUTEX_LOCK(&Mutex[2]); /* lock */
                                        Event[2]++;
                                        COND_SIGNAL(&Condition[2]);
                                MUTEX_UNLOCK(&Mutex[2]); /* unlock */
                        break;
        }
        GlobalCounter++;
}

void *thread_main0(void *arg)
{
        int class = 3;                                          /* FIFO Schedule */
        int priority = 1;                                       /* RealTime Priority min */
        struct sched_param param;       /* RealTime Priority min */
        cpuset_t    *cpumask;
        cpu_t                   cpu = 1;
        volatile int thread_no;
        int i,sig;
        float min,max,avr;
        struct timespec w0 = {0,100 * 1000* 1000};
        struct timespec wait = {0,200* 1000};
        sigset_t set;

            sigemptyset(&set);
            sigaddset(&set,SIGRTMIN+2);
            sigaddset(&set,SIGINT);
        if (pthread_getschedparam(pthread_self(),&class,&param))
        {
                        fprintf(stderr, "pthread %d Cannot get priority %s¥n",
                        pthread_self(),strerror(errno));
```

```
        }
        param.sched_priority = priority;
        if (pthread_setschedparam(pthread_self(),SCHED_FIFO,(const struct sched_param *)&param))
        {
                        fprintf(stderr, "pthread %d Cannot set priority %s¥n",
                        pthread_self(),strerror(errno));
        }
#ifdef CCURRT
        cpumask = cpuset_alloc();
        cpuset_init(cpumask);
        cpuset_set_cpu(cpumask,cpu,1);/* 2nd CPU set */
        if ( mpadvise(MPA_PRC_SETBIAS,MPA_TID,0,cpumask)<0)
        {
                        fprintf(stderr, "thread %d Cannot bind processor %s¥n",
                        pthread_self(),strerror(errno));
        }
            if (mpadvise (MPA_PRC_GETRUN,MPA_TID,0,cpumask)!=MPA_FAILURE)
            {
                    char * cpustr;
                      cpustr = cpuset_get_string(cpumask);
                      printf("Thread %d is ¥"%s¥" running¥n",*(int*)arg,cpustr);
                      free(cpustr);
            }
        cpuset_free(cpumask);

#endif
        MUTEX_LOCK(&Mutex[0]);
        Event[0]++;
        MUTEX_UNLOCK(&Mutex[0]);
        do
        {
                        MUTEX_LOCK(&Mutex[0]);
                        thread_no = Event[0];
                        MUTEX_UNLOCK(&Mutex[0]);
                        nanosleep(&w0,0);
        } while (thread_no<THREAD_NO);
            if (pthread_sigmask(SIG_SETMASK,&eiset,&oset)==(-1))
            {
                        fprintf(stderr, "Cannot set sigprocmask - %s¥n",
                        strerror(errno));
        }
            do
            {
                        //sigwait(&set,&sig);interval_hadler(sig);
                        sigsuspend(&eiset);
                        if(GlobalCounter>=TESTNUMBER) loop_flag=0;
            } while (loop_flag);
        COND_SIGNAL(&Condition[1]);
        COND_SIGNAL(&Condition[2]);
        min=99999999999.0,max=0,avr=0;
        for(i=0;i<TESTNUMBER/2;i++)
        {
                        if(min>SwitchTime1[i]) min = SwitchTime1[i];
                        if(max<SwitchTime1[i]) max = SwitchTime1[i];
                        avr += SwitchTime1[i];
        }
        printf("Switch1 %10.3f: %10.3f : %10.3f¥t",min,avr/(float)(TESTNUMBER/2),max);
            if (fabs(max-min)<=MAXJITTER)
            {
                        printf("GOOD(fabs(%f)<%d micro sec)¥n",max-min,MAXJITTER);
            }
            else
            {
                        printf("FAIL(fabs(%f)>%d micro sec)¥n",max-min,MAXJITTER);
            }
```

```
	min=99999999999.0,max=0,avr=0;
	for(i=0;i<TESTNUMBER/2;i++)
	{
		if(min>SwitchTime2[i]) min = SwitchTime2[i];
		if(max<SwitchTime2[i]) max = SwitchTime2[i];
		avr += SwitchTime2[i];
	}
	printf("Switch2 %10.3f: %10.3f : %10.3f¥t",min,avr/(float)(TESTNUMBER/2),max);
	    if (fabs(max-min)<=MAXJITTER)
	    {
		printf("GOOD(fabs(%f)<%d micro sec)¥n",max-min,MAXJITTER);
	    }
	    else
	    {
		printf("FAIL(fabs(%f)>%d micro sec)¥n",max-min,MAXJITTER);
	    }

	min=99999999999.0,max=0,avr=0;
	for(i=1;i<TESTNUMBER;i++)
	{
		if(min>IntervalTime[i]) min = IntervalTime[i];
		if(max<IntervalTime[i]) max = IntervalTime[i];
		avr += IntervalTime[i];
	}
	printf("Interval %10.3f: %10.3f : %10.3f¥t",min,avr/(float)(TESTNUMBER-1),max);
	    if (fabs(max-min)<=MAXJITTER)
	    {
		printf("GOOD(fabs(%f)<%d micro sec)¥n",max-min,MAXJITTER);
	    }
	    else
	    {
		printf("FAIL(fabs(%f)>%d micro sec)¥n",max-min,MAXJITTER);
	    }


	return(0);
}


void *thread_main1(void *arg)
{
	static int err=-1;
	int class = 3;				/* FIFO Schedule */
	int priority = 2;					/* RealTime Priority min */
	static struct sched_param param; /* RealTime Priority min */
	cpuset_t    *cpumask;
	cpu_t cpu = 1;							/* boot PROCESSOR */

	if (pthread_getschedparam(pthread_self(),&class,&param))
	{
		fprintf(stderr, "thread %d Cannot get priority %s¥n",
		pthread_self(),strerror(errno));
	}
	param.sched_priority = priority;
	if (pthread_setschedparam(pthread_self(),SCHED_FIFO,(const struct sched_param *)&param))
	{
		fprintf(stderr, "thread %d Cannot set priority %s¥n",
		pthread_self(),strerror(errno));
	}
	    if (pthread_sigmask(SIG_SETMASK,&diset,&oset)==(-1))
	    {
		fprintf(stderr, "Cannot set sigprocmask - %s¥n",
		strerror(errno));
		err = errno;
```

```
                pthread_exit((void*)&err);
        }
#ifdef CCURRT
        cpumask = cpuset_alloc();
        cpuset_init(cpumask);
        cpuset_set_cpu(cpumask,cpu,1);/* CPU set */
        if ( mpadvise(MPA_PRC_SETBIAS,MPA_TID,0,cpumask)<0)
        {
                fprintf(stderr, "thread %d Cannot bind processor %s¥n",
                pthread_self(),strerror(errno));
        }
            if (mpadvise (MPA_PRC_GETRUN,MPA_TID,0,cpumask)!=MPA_FAILURE)
            {
                char * cpustr;
                cpustr = cpuset_get_string(cpumask);
                printf("Thread %d is ¥"%s¥" running¥n",*(int*)arg,cpustr);
                free(cpustr);
            }
        cpuset_free(cpumask);
#endif
        MUTEX_LOCK(&Mutex[0]);
        Event[0]++;
        MUTEX_UNLOCK(&Mutex[0]);
            do
            {
                MUTEX_LOCK(&Mutex[1]); /* lock */
                        if(Event[1]==0) COND_WAIT(&Condition[1],&Mutex[1]);
                        readtime(&EndTime);
                        if (GlobalCounter<TESTNUMBER)
adjust(&StartTime,&EndTime,&SwitchTime1[GlobalCounter/2]);
                        Event[1]--;
                MUTEX_UNLOCK(&Mutex[1]); /* unlock */
                /* event1_exec(); */
            } while (loop_flag);



        err=0;
            pthread_exit((void *)err);
        return(0);
}
void *thread_main2(void *arg)
{
        static int err=-1;
        int class = 3;                                          /* FIFO Schedule */
        int priority = 3;                                   /* RealTime Priority min */
        struct sched_param param;          /* RealTime Priority min */
        cpuset_t    *cpumask;
        cpu_t cpu = 1;                                          /* 2nd PROCESSOR */

        if (pthread_getschedparam(pthread_self(),&class,&param))
        {
                fprintf(stderr, "thread %d Cannot get priority %s¥n",
                pthread_self(),strerror(errno));
        }
        param.sched_priority = priority;
        if (pthread_setschedparam(pthread_self(),SCHED_FIFO,(const struct sched_param *)&param))
        {
                fprintf(stderr, "thread %d Cannot set priority %s¥n",
                pthread_self(),strerror(errno));
        }
            if (pthread_sigmask(SIG_SETMASK,&diset,&oset)==(-1))
            {
                fprintf(stderr, "Cannot set sigprocmask - %s¥n",
                strerror(errno));
```

```
                        err = errno;
                          pthread_exit((void*)&err);
                }
#ifdef CCURRT
        cpumask = cpuset_alloc();
        cpuset_init(cpumask);
        cpuset_set_cpu(cpumask,cpu,1);/* CPU set */
        if ( mpadvise(MPA_PRC_SETBIAS,MPA_TID,0,cpumask)<0)
        {
                        fprintf(stderr, "thread %d Cannot bind processor %s¥n",
                        pthread_self(),strerror(errno));
        }
                if (mpadvise (MPA_PRC_GETRUN,MPA_TID,0,cpumask)!=MPA_FAILURE)
                {
                        char * cpustr;
                          cpustr = cpuset_get_string(cpumask);
                          printf("Thread %d is ¥"%s¥" running¥n",*(int*)arg,cpustr);
                          free(cpustr);
                }
        cpuset_free(cpumask);
#endif
                MUTEX_LOCK(&Mutex[0]);
                Event[0]++;
                MUTEX_UNLOCK(&Mutex[0]);
                do
                {
                        MUTEX_LOCK(&Mutex[2]); /* lock */
                                if (Event[2]==0) COND_WAIT(&Condition[2],&Mutex[2]);
                                readtime(&EndTime);
                                if (GlobalCounter<TESTNUMBER)
adjust(&StartTime,&EndTime,&SwitchTime2[GlobalCounter/2]);
                                Event[2]--;
                        MUTEX_UNLOCK(&Mutex[2]); /* unlock */
                        /* event2_exec(); */
                } while (loop_flag);


           pthread_exit((void *)err);
        return(0);
}

void       *(*thread_main[THREAD_NO])(void *arg) =
{
        thread_main0,
        thread_main1,
        thread_main2
};



do_main(progname,fdin,filename,opt)
                char       *progname;
                FILE       *fdin;
                char       *filename;
        int opt;
{
        static int    no,thread_status[THREAD_NO],thread_arg[THREAD_NO];
        static pthread_t          tid[THREAD_NO];
        static pthread_attr_t     attributes[THREAD_NO];
        void *status_p;
        int signum;
                static struct sigaction newact;
                static sigset_t set,oset;
        static timer_t timer_id;
```

```
printf("************************************************** ¥n");
printf("**** posix timer & thread contexst switch test **** ¥n");
printf("************************************************** ¥n");


        if (opt)
        {
                system("/usr/bin/shield -a 1");
                system("/usr/bin/shield -c");
        }

            if (sigprocmask(0,NULL,&set)==(-1))
            {
                fprintf(stderr, "%s: Cannot get sigprocmask - %s¥n",
                progname,strerror(errno));
                exit(1);
            }
            sigaddset(&set,SIGRTMIN+2);
            if (sigprocmask(SIG_SETMASK,&set,&oset)==(-1))
            {
                fprintf(stderr, "%s: Cannot set sigprocmask - %s¥n",
                progname,strerror(errno));
                exit(1);
            }
        signum = SIGRTMIN+2;
        printf("Create POSIX timer %d Hz¥n",CYCLE);
        timer_id = create_posix_timer(signum,0,1000000000/CYCLE);
        if (timer_id<0)
        {
                                fprintf(stderr, "%s: Cannot set posix timer - %s¥n",
                progname,strerror(errno));
                exit(1);
        }
            /* Set up the signal handler using sigaction(2) or sigset(2) */
            sigemptyset(&newact.sa_mask);
            sigaddset(&newact.sa_mask,signum);
            newact.sa_handler = interval_hadler;
            newact.sa_flags = SA_SIGINFO|SA_RESTART;
            if (sigaction(signum, &newact, 0)==(-1))
            {
                fprintf(stderr, "%s: Cannot set sigaction - %s¥n",
                progname,strerror(errno));
                exit(1);
            }

            sigfillset(&diset);
        sigfillset(&eiset);
        sigdelset(&eiset,SIGRTMIN+2);
        sigdelset(&eiset,SIGALRM);
        sigdelset(&eiset,SIGINT);

        printf("Create %d thread¥n",THREAD_NO);
        for (no=0;no<THREAD_NO;no++)
        {
                if (pthread_cond_init(&Condition[no],0)<0)
                {
                                fprintf(stderr, "%s: Cannot create condition - %s¥n",
                                progname,strerror(errno));
                                exit(1);
                }
                if (pthread_mutex_init(&Mutex[no],0)<0)
                {
                                fprintf(stderr, "%s: Cannot create condition - %s¥n",
                                progname,strerror(errno));
                                exit(1);
```

```
                }
                Event[no]=0;
        }
        for (no=0;no<THREAD_NO;no++)
        {
                if (pthread_attr_init(&attributes[no]))
                {
                        fprintf(stderr, "%s: Cannot set attributes - %s¥n",
                        progname,strerror(errno));
                        exit(1);
                }
                  if (pthread_attr_setscope(&attributes[no],PTHREAD_SCOPE_SYSTEM))
                {
                        fprintf(stderr, "%s: Cannot set attributes - %s¥n",
                        progname,strerror(errno));
                        exit(1);
                }
                  if (pthread_attr_setdetachstate(&attributes[no],PTHREAD_CREATE_JOINABLE))
                {
                        fprintf(stderr, "%s: Cannot set attributes - %s¥n",
                        progname,strerror(errno));
                        exit(1);
                }
                  if (pthread_attr_setinheritsched(&attributes[no],PTHREAD_INHERIT_SCHED))
                {
                        fprintf(stderr, "%s: Cannot set attributes - %s¥n",
                        progname,strerror(errno));
                        exit(1);
                }
        }
        MUTEX_LOCK(&Mutex[0]); /* lock */
        for (no=1;no<THREAD_NO;no++)
        {
                thread_arg[no] = no;
                  thread_status[no] = pthread_create(&tid[no],&attributes[no],thread_main[no],(void *)&thread_arg[no]);
                if (thread_status[no])
                {
                        fprintf(stderr, "%s: Cannot create thread - %s¥n",
                        progname,strerror(errno));
                        exit(1);
                }
        }
        MUTEX_UNLOCK(&Mutex[0]); /* unlock */
        thread_arg[0] = 0;
        thread_main[0]((void *)&thread_arg[0]);
        timer_delete(timer_id);
        for (no=1;no<THREAD_NO;no++)
        {
                pthread_join(tid[no],&status_p);
        }
}


main(argc, argv)
        int argc;
        char **argv;
{
        extern int errno;
        extern int optind;
        extern char *optarg;
        register int i,c, opt = 0;

        progname = argv[0];
        while((c = getopt(argc, argv, "b")) != EOF){
                switch(c) {
```

```
                        case 'b':
                                opt =1;
                                break;
                        default:
                                fprintf(stderr,
                                "usage: %s [-b]¥n",
                                progname);
                                exit(1);
                }
        }
        if (optind == argc) {
                do_main(progname,stdin, "(stdin)",opt);
        } else {
                register FILE *fd;
                for (c = optind; c < argc; c++) {
                        if ((fd = fopen(argv[c], "w")) == NULL) {
                                fprintf(stderr,
                               "%s: ¥"%s¥" %s¥n",
                                  progname,argv[c],strerror(errno));
                                exit(1);
                        }
                        do_main(progname,fd,argv[c],opt);
                        fprintf(fd,"Interval,Switch1,Switch2¥n");
                        for(i=0;i<TESTNUMBER/2;i++)
                        {
                                fprintf(fd,"%f,%f,%f¥n",IntervalTime[i+1], SwitchTime1[i], SwitchTime2[i]);
                        }
                        for(i=TESTNUMBER/2;i<TESTNUMBER-1;i++)
                        {
                                fprintf(fd,"%f,,¥n",IntervalTime[i+1]);
                        }
                        fclose(fd);
                }
        }
}
```

SSE/SSE2 命令
gcc では生成されない。ため、インラインアセンブラを使用する。

例題

```
        for(i=0;i<32;i++)
        {
                v2[0] += v0[i] * v1[i];
        }
```

SSE/SSE2 例

```
//ベクタ型を宣言(メモリのアライメントに注意)
typedef struct { double v[32]; } vector _attribute_ ((aligned (16)));;

        vector v0,v1,v2;
        _inline_sse_add_mul_vector(v2,v0,v1);//インライン展開する

//アセンブラで記述2×2データ毎8回計算する
#define _inline_sse_add_mul_vector(v0,v1,v2) ¥
{ ¥     //データ0にデータ0－4の計算結果が残る
_asm_ _volatile_ ("prefetcht0 %0 ¥n¥t" ¥                    //プリフェッチ
                        "prefetcht0 %4 ¥n¥t" ¥              //プリフェッチ
                        "movapd %0, %%xmm0 ¥n¥t" ¥                  //データ0代入 V0(v[0],v[1])
                        "movapd %1, %%xmm1 ¥n¥t" ¥                  //データ1代入 V0(v[2],v[3])
                        "movapd %2, %%xmm2 ¥n¥t" ¥                  //データ2代入 V0(v[4],v[5])
                        "movapd %3, %%xmm3 ¥n¥t" ¥                  //データ3代入 V0(v6[],v[7])
                        "movapd %4, %%xmm4 ¥n¥t" ¥                  //データ4代入 V1(v[0],v[1])
                        "movapd %5, %%xmm5 ¥n¥t" ¥                  //データ5代入 V2(v[2],v[3])
                        "movapd %6, %%xmm6 ¥n¥t" ¥                  //データ6代入 V2(v[4],v[5])
                        "movapd %7, %%xmm7 ¥n¥t" ¥                  //データ7代入 V2(v[6],v[7])
                        "mulpd %%xmm4, %%xmm0 ¥n¥t" ¥               //データ0＝データ4×データ0
                        "mulpd %%xmm5, %%xmm1 ¥n¥t" ¥               //データ1＝データ5×データ1
                        "mulpd %%xmm6, %%xmm2 ¥n¥t" ¥               //データ2＝データ6×データ2
                        "mulpd %%xmm7, %%xmm3 ¥n¥t" ¥               //データ3＝データ7×データ3
                        "addpd %%xmm1, %%xmm0 ¥n¥t" ¥               //データ0＝データ1＋データ0
                        "addpd %%xmm3, %%xmm2 ¥n¥t" ¥               //データ2＝データ3＋データ2
                        "addpd %%xmm2, %%xmm0 ¥n¥t" ¥               //データ0＝データ2＋データ0 加算結果は
xmm0
                        : ¥
                        : ¥
                        "m" ((v0)->v[0]),   ¥                       //データ0宣言
                        "m" ((v0)->v[2]),   ¥                       //データ1宣言
                        "m" ((v0)->v[4]),   ¥                       //データ2宣言
                        "m" ((v0)->v[6]),   ¥                       //データ3宣言
                        "m" ((v1)->v[0]),   ¥                       //データ4宣言
                        "m" ((v1)->v[2]),   ¥                       //データ5宣言
                        "m" ((v1)->v[4]),   ¥                       //データ6宣言
                        "m" ((v1)->v[6])); ¥                        //データ7宣言
_asm_ _volatile_ ("movapd %0, %%xmm1 ¥n¥t" ¥                //同様に V[8]から V[13]まで計算する
                        "movapd %1, %%xmm2 ¥n¥t" ¥
                        "movapd %2, %%xmm3 ¥n¥t" ¥
                        "movapd %3, %%xmm5 ¥n¥t" ¥
                        "movapd %4, %%xmm6 ¥n¥t" ¥
                        "movapd %5, %%xmm7 ¥n¥t" ¥
                        "mulpd %%xmm5, %%xmm1 ¥n¥t" ¥
                        "mulpd %%xmm6, %%xmm2 ¥n¥t" ¥
                        "mulpd %%xmm7, %%xmm3 ¥n¥t" ¥
                        "addpd %%xmm1, %%xmm0 ¥n¥t" ¥               //加算結果は xmm0
                        "addpd %%xmm3, %%xmm2 ¥n¥t" ¥
                        "addpd %%xmm2, %%xmm0 ¥n¥t" ¥               //加算結果は xmm0
                        : ¥
                        : ¥
                        "m" ((v0)->v[8]),   ¥
                        "m" ((v0)->v[10]),   ¥
                        "m" ((v0)->v[12]),   ¥
                        "m" ((v1)->v[8]),   ¥
                        "m" ((v1)->v[10]),   ¥
                        "m" ((v1)->v[12])); ¥
_asm_ _volatile_ ("movapd %0, %%xmm1 ¥n¥t" ¥                        //同様に V[14]から V[19]まで
                        "movapd %1, %%xmm2 ¥n¥t" ¥
                        "movapd %2, %%xmm3 ¥n¥t" ¥
                        "movapd %3, %%xmm5 ¥n¥t" ¥
                        "movapd %4, %%xmm6 ¥n¥t" ¥
                        "movapd %5, %%xmm7 ¥n¥t" ¥
```

```
                        "mulpd %%xmm5, %%xmm1 ¥n¥t" ¥
                        "mulpd %%xmm6, %%xmm2 ¥n¥t" ¥
                        "mulpd %%xmm7, %%xmm3 ¥n¥t" ¥
                        "addpd %%xmm1, %%xmm0 ¥n¥t" ¥           //加算結果は xmm0
                        "addpd %%xmm3, %%xmm2 ¥n¥t" ¥
                        "addpd %%xmm2, %%xmm0 ¥n¥t" ¥           //加算結果は xmm0
                        : ¥
                        : ¥
                        "m" ((v0)->v[14]),   ¥
                        "m" ((v0)->v[16]),   ¥
                        "m" ((v0)->v[18]),   ¥
                        "m" ((v1)->v[14]),   ¥
                        "m" ((v1)->v[16]),   ¥
                        "m" ((v1)->v[18])); ¥
_asm_ _volatile_ ("movapd %0, %%xmm1 ¥n¥t" ¥            //同様に V[20]から V[25]まで
                        "movapd %1, %%xmm2 ¥n¥t" ¥
                        "movapd %2, %%xmm3 ¥n¥t" ¥
                        "movapd %3, %%xmm5 ¥n¥t" ¥
                        "movapd %4, %%xmm6 ¥n¥t" ¥
                        "movapd %5, %%xmm7 ¥n¥t" ¥
                        "mulpd %%xmm5, %%xmm1 ¥n¥t" ¥
                        "mulpd %%xmm6, %%xmm2 ¥n¥t" ¥
                        "mulpd %%xmm7, %%xmm3 ¥n¥t" ¥
                        "addpd %%xmm1, %%xmm0 ¥n¥t" ¥           //加算結果は xmm0
                        "addpd %%xmm3, %%xmm2 ¥n¥t" ¥
                        "addpd %%xmm2, %%xmm0 ¥n¥t" ¥           //加算結果は xmm0
                        : ¥
                        : ¥
                        "m" ((v0)->v[20]),   ¥
                        "m" ((v0)->v[22]),   ¥
                        "m" ((v0)->v[24]),   ¥
                        "m" ((v1)->v[20]),   ¥
                        "m" ((v1)->v[22]),   ¥
                        "m" ((v1)->v[24])); ¥
_asm_ _volatile_ ("movapd %0, %%xmm1 ¥n¥t" ¥            //同様に V[26]から V[31]まで
                        "movapd %1, %%xmm2 ¥n¥t" ¥
                        "movapd %2, %%xmm3 ¥n¥t" ¥
                        "movapd %3, %%xmm5 ¥n¥t" ¥
                        "movapd %4, %%xmm6 ¥n¥t" ¥
                        "movapd %5, %%xmm7 ¥n¥t" ¥
                        "mulpd %%xmm5, %%xmm1 ¥n¥t" ¥
                        "mulpd %%xmm6, %%xmm2 ¥n¥t" ¥
                        "mulpd %%xmm7, %%xmm3 ¥n¥t" ¥
                        "addpd %%xmm1, %%xmm0 ¥n¥t" ¥           //加算結果は xmm0
                        "addpd %%xmm3, %%xmm2 ¥n¥t" ¥
                        "addpd %%xmm2, %%xmm0 ¥n¥t" ¥           //加算結果は xmm0
                        : ¥
                        : ¥
                        "m" ((v0)->v[26]),   ¥
                        "m" ((v0)->v[28]),   ¥
                        "m" ((v0)->v[30]),   ¥
                        "m" ((v1)->v[26]),   ¥
                        "m" ((v1)->v[28]),   ¥
                        "m" ((v1)->v[30])); ¥
_asm_ _volatile_ ("movapd %%xmm0, %%xmm4 ¥n¥t" ¥        //xmm0 のデータを xmm4 に移動
                        "unpckhpd %%xmm4, %%xmm4 ¥n¥t" ¥        //xmm4 のデータを2つに分解
                        "addpd %%xmm4, %%xmm0 ¥n¥t" ¥           //2つに分解したデータを加算
                          "movlpd %%xmm0, %0 ¥n¥t" ¥            //xmm0 の答えを戻す
                        : ¥
                        "=m" ((v2)->v[0]));¥
}
```