Chapter 8. Signal

Generate Signal

- kill (시그널 전송 명령어)

kill -S VSR1 pid
시그널 이름으로 바꿀다.

- kill (pid, sig)
↓
if 0 : 같은 그룹의 모든 프로세스에 전송
if -1 : 가능한 모든 프로세스에 전달
if 음수 : 절댓값과 같은 그룹에 전달

- raise (sig)
나 자신에게 전달

- alarm (second)
second 후에 SIGALRM을 자신에게 전달
second가 0이면 이전 alarm 취소

Signal set

- sig.addset (sig_set, sig_no)
sig_set에 sig_no 시그널 추가.

- sigdelset (sig_set, sig_no)

  sig_set에 sig_no 시그널 있으면 삭제.

- sigemptyset (sig_set)

  sig_set 비우기.

- sigfillset (sig_set)

  sig_set에 가능한 모든 시그널을 추가.

- sigismember (sig_set, sig_no)

  sig_no 시그널이 sig_set에 있으면 1, 아니면 0 반환

Signal Mask

sigprocmask (how, sig_set, old_set)

→ SIG_BLOCK : sig_set의 시그널들을 MASK에 추가

- SIG_UNBLOCK : sig_set의 시그널들을 mask에서 삭제
- SIG_SETMASK : sig_set 자체를 mask로 변경

Old_set 에는 변경 전 mask 가 저장되어 나옴.

☆ single thread 에서만 사용!

SIGSTOP, SIGKILL은 이것으로 못막음.

Catching Signal

- sig_action (Signal, Sig action, old - action)

Struct sigaction
- handler (void 리턴에 인자로 int 받는 함수) or SIG_DFL, SIG_IGN
  - 기본 / 무시
- handler가 실행중일때 block 할 시그널들의 sig_set
- flag 및 옵션 (realtime handler를 사용할지 말지)
- handler 만으로 부족할 때 여기에 realtime handler 등록

Signal handler 안은 critical section

↓

volatile 키워드로 메모리 직접 참조하도록.

sig_atomic_t 타입으로 atomic 하게 access.

critical section 안의 변수 값을 읽거나 쓸 때 mask로 막고, 다 끝나면 되돌리기.

Wait for Signal

- pause (void)

  호출한 thread를 suspend. 시그널이 도착하면 리턴됨. (항상 -1)

flag sig_received = 0

while (sig_received == 0) ← 여기서 값 비교하고, pause 부르기 전에

  pause() 시그널이 온다면 오류 가능.

그래서 sigsuspend (sig_set)

sig_set을 mask로 설정함과 동시에 시그널을 기다림.

아니 원하는 시그널을 mask에 넣어 막음. ※
그 다음 sigsuspend (원하는 시그널이 없는 sig_set)으로 원하는 시그널을 열면서 기다린다.

- sigwait (sigmask, signo)

일단 block 하고, sigmask에 있는 signal이 pending되면,
그 시그널을 pending list에서 지우고 그 시그널 번호가 signo

sigwait 전에 sigmask에 있는 시그널을 block 하고 sigwait해야함.

- sigsetjmp (env, savemask), siglongjmp (env, val)
  |
  0 반환이면 설정완료, 다른 값이면 점프해서온 것 (val 값)

siglongjmp를 만나면 sigsetjmp가 불렸던 곳을 점프.

Asyncronous I/O

I/O는 백그라운드에서 돌게하고, 프로세스는 다른 일을 하게 함.
aio 요청시, queue에 들어가고, 바로 0을 반환