

---

# Linux System Programming #6-2

## Lecture Notes

**Spring 2020**

**School of Computer Science and Engineering,**

**Soongsil University, Seoul, Korea**

**Jiman Hong**

**[jiman@acm.org](mailto:jiman@acm.org)**

# sigprocmask(2)

```
#include <signal.h>
int sigprocmask(int how, const sigset_t *set, sigset_t *oldset);
```

리턴 값 : 성공 시 0, 에러 시 -1을 리턴하고 errno가 설정됨

- 현재의 시그널 마스크를 검사하거나 변경하는 시스템호출 함수
- 프로세스의 시그널 마스크
  - 시그널이 마스크(블록)되어 있어 현재 프로세스에게 전달되지 않는 시그널들의 집합
- how
  - 시그널 마스크의 구체적인 변경 방법을 정의하는 상수
  - 시그널 마스크를 설정 => set 인자에 NULL 포인터가 아닌 포인터를 넣고 시그널 마스크의 구체적인 변경 방법을 how 인자에 정의
- set
  - 마스크하거나 마스크를 해제할 시그널 집합의 주소
  - set 인자가 NULL이라면, 시그널 마스크는 변경되지 않고 how 인자가 어떤 값을 가지고 있어도 무시
- oldset
  - 이전 시그널 설정 값을 저장할 시그널 집합의 주소, NULL. 사용 가능
  - 시그널 마스크를 검사할 때 oldset 인자에 NULL 포인터가 아닌 값을 넣으면 프로세스의 현재 시그널 마스크가 oldset 인자가 가리키는 곳에 저장

## how 인자에 따른 sigprocmask()의 실행

how 인자	설명
SIG_BLOCK	프로세스의 현재 시그널 마스크와 set이 가리키는 시그널 집합의 합집합이 프로세스의 새 프로세스 마스크가 됨. 주로 블록할 시그널들을 더 추가할 때 유용
SIG_UNBLOCK	프로세스의 현재 시그널 마스크와 set이 가리키는 시그널 집합의 교집합이 프로세스의 새 프로세스 마스크가 됨. 주로 블록된 시그널들에 대해 블록을 해제할 때 유용
SIG_SETMASK	이전 블록된 시그널 집합을 모두 지우고 set이 가리키는 시그널 집합의 값이 프로세스의 새 시그널 마스크로 설정

# sigprocmask() 예제

```
<ssu_sigprocmask.c>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>

int main(void)
{
    sigset_t sig_set;
    int count;

    sigemptyset(&sig_set);
    sigaddset(&sig_set, SIGINT);
    sigprocmask(SIG_BLOCK, &sig_set, NULL);

    for (count = 3 ; 0 < count ; count--) {
        printf("count %d\n", count);
        sleep(1);
    }

    printf("Ctrl-C에 대한 블록을 해제\n");
    sigprocmask(SIG_UNBLOCK, &sig_set, NULL);
    printf("count중 Ctrl-C입력하면 이 문장은 출력 되지 않음.\n");

    while (1);

    exit(0);
}
```

## 실행 결과

```
root@localhost:/home/oslab# ./ssu_sigprocmask
count 3
count 2
count 1
Ctrl-C에 대한 블록을 해제
count중 Ctrl-C입력하면 이 문장은 출력 되지 않음.
^C
root@localhost:/home/oslab# ./ssu_sigprocmask
count 3
^Ccount 2
^Ccount 1
Ctrl-C에 대한 블록을 해제
```

# sigpending(2)

```
#include <signal.h>
```

```
int sigpending(sigset_t *set);
```

리턴 값 : 성공 시 0, 에러 시 -1을 리턴하고 errno가 설정됨

- sigpending()
  - 팬딩 중이거나, 블록되었을 때 발생한 시그널 집합을 가져오는 시스템호출 함수
  - 시그널을 블록된 상태에서 어떤 시그널이 발생해서 블록되었는지 확인
  - set 인자가 가리키는 곳에 호출한 프로세스에 시그널이 블록되어 있으며 현재 팬딩 중인 시그널들의 집합을 저장

# sigpending() 예제

```
<ssu_sigpending.c>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>

int main(void)
{
    sigset_t pendingset;
    sigset_t sig_set;
    int count = 0;

    sigfillset(&sig_set);
    sigprocmask(SIG_SETMASK, &sig_set, NULL);

    while (1) {
        printf("count: %d\n", count++);
        sleep(1);

        if (sigpending(&pendingset) == 0) {
            if (sigismember(&pendingset, SIGINT)) {
                printf("SIGINT가 블록되어 대기 중. 무한 루프를 종료.\n");
                break;
            }
        }
    }

    exit(0);
}
```

## 실행 결과

```
root@localhost:/home/oslab# ./ssu_sigpending
count: 0
count: 1
count: 2
count: 3
^Zcount: 4
^Zcount: 5
count: 6
count: 7
count: 8
^CSIGINT가 블록되어 대기 중. 무한 루프를 종료.
```

# sigaction(2)

```
#include <signal.h>
int sigaction(int signo, const struct sigaction *act, struct sigaction *oldact);
```

리턴 값 : 성공 시 0, 에러 시 -1을 리턴하고 errno가 설정됨

- sigpending()는 팬딩 중이거나, 블록되었을 때 발생한 시그널 집합을 가져오는 시스템호출 함수이다. set 인자가 가리키는 곳에 호출한 프로세스에 시그널이 블록되어 있으며 현재 팬딩 중인 시그널들의 집합을 저장한다.
- act : 액션 수정
- oldact : 이전 액션

```
struct sigaction{
    void (*sa_handler)(int);      //SIG_IGN나 SIG_DFL의 시그널 핸들러 주소
    sigset_t sa_mask;            //추가적으로 블록할 시그널
    int sa_flags;                 //시그널 옵션, <표 6-10> 참조
    void (*sa_sigaction)(int, siginfo_t*, void*); //대안 핸들러
};
```

# 각 시그널 처리에 대한 옵션 플래그들(sa\_flags)

옵션	설명
SA_INTERRUPT	이 시그널에 의해 가로채인 시스템 호출이 자동으로 재시작 되지 않음
SA_NOCLDSTOP	signo가 SIGCHLD인 경우, 자식 프로세스가 중단되었을 때 이 시그널을 발생하지 않도록 함
SA_NOCLDWAIT	signo가 SIGCHLD인 경우, 이 옵션은 호출한 프로세스의 자식 프로세스들이 종료되었을 때 시스템이 좀비 프로세스들을 만들지 못하게 함
SA_NODEFER	이 시그널에 의해 시그널 핸들러가 실행되는 도중에 시스템이 같은 시그널을 자동으로 블록되지 않게 함
SA_ONSTACK	sigaltstack(2)로 대안 스택이 선언되어 있다면 시그널이 대안 스택의 프로세스에 전달되게 함
SA_RESETHAND	시그널 핸들러에 진입할 때 시그널의 처분 방식을 SIG_DFL로 재설정하고 SA_SIGINFO플래그를 해제함
SA_RESTART	이 시그널에 의해 가로채인 시스템 호출이 자동으로 재시작하게 함
SA_SIGINFO	시그널 핸들러에 추가적인 정보 두 개를 전달하게 함



# sigaction() 예제 1

```
<ssu_sigaction_1.c>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>

void ssu_signal_handler(int signo) {
    printf("ssu_signal_handler control\n");
}

int main(void) {
    struct sigaction sig_act;
    sigset_t sig_set;

    sigemptyset(&sig_act.sa_mask);
    sig_act.sa_flags = 0;
    sig_act.sa_handler = ssu_signal_handler;
    sigaction(SIGUSR1, &sig_act, NULL);
    printf("before first kill()\n");
    kill(getpid(), SIGUSR1);
    sigemptyset(&sig_set);
    sigaddset(&sig_set, SIGUSR1);
    sigprocmask(SIG_SETMASK, &sig_set, NULL);
    printf("before second kill()\n");
    kill(getpid(), SIGUSR1);
    printf("after second kill()\n");
    exit(0);
}
```

## 실행 결과

```
root@localhost:/home/oslab# ./ssu_sigaction_1
before first kill()
ssu_signal_handler control
before second kill()
after second kill()
```

## sigaction() 예제 2

```
<ssu_sigaction_2.c>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>

void ssu_check_pending(int signo, char *signame);
void ssu_signal_handler(int signo);

int main(void)
{
    struct sigaction sig_act;
    sigset_t sig_set;

    sigemptyset(&sig_act.sa_mask);
    sig_act.sa_flags = 0;
    sig_act.sa_handler = ssu_signal_handler;

    if (sigaction(SIGUSR1, &sig_act, NULL) != 0) {
        fprintf(stderr, "sigaction() error\n");
        exit(1);
    }
    else {
        sigemptyset(&sig_set);
        sigaddset(&sig_set, SIGUSR1);

        if (sigprocmask(SIG_SETMASK, &sig_set, NULL) != 0) {
            fprintf(stderr, "sigprocmask() error\n");
            exit(1);
        }
        else {
            printf("SIGUSR1 signals are now blocked\n");
        }
    }
}
```

```
kill(getpid(), SIGUSR1);
printf("after kill()\n");
ssu_check_pending(SIGUSR1, "SIGUSR1");
sigemptyset(&sig_set);
sigprocmask(SIG_SETMASK, &sig_set, NULL);
printf("SIGUSR1 signals are no longer blocked\n");
ssu_check_pending(SIGUSR1, "SIGUSR1");
}
}
exit(0);
}

void ssu_check_pending(int signo, char *signame) {
    sigset_t sig_set;

    if (sigpending(&sig_set) != 0)
        printf("sigpending() error\n");
    else if (sigismember(&sig_set, signo))
        printf("a %s signal is pending\n", signame);
    else
        printf("%s signals are not pending\n", signame);
}

void ssu_signal_handler(int signo) {
    printf("in ssu_signal_handler function\n");
}

실행 결과
root@localhost:/home/oslab# ./ssu_sigaction_2
SIGUSR1 signals are now blocked
after kill()
a SIGUSR1 signal is pending
in ssu_signal_handler function
SIGUSR1 signals are no longer blocked
SIGUSR1 signals are not pending
```

## sigaction() 예제 3

```
<ssu_sigaction_3.c>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
static void ssu_signal_handler1(int signo);
static void ssu_signal_handler2(int signo);

int main(void)
{
    struct sigaction act_int, act_quit;

    act_int.sa_handler = ssu_signal_handler1;
    sigemptyset(&act_int.sa_mask);
    sigaddset(&act_int.sa_mask, SIGQUIT);
    act_quit.sa_flags = 0;

    if (sigaction(SIGINT, &act_int, NULL) < 0) {
        fprintf(stderr, "sigaction(SIGINT) error\n");
        exit(1);
    }
    act_quit.sa_handler = ssu_signal_handler2;
    sigemptyset(&act_quit.sa_mask);
    sigaddset(&act_quit.sa_mask, SIGINT);
    act_int.sa_flags = 0;
    if (sigaction(SIGQUIT, &act_quit, NULL) < 0) {
        fprintf(stderr, "sigaction(SIGQUIT) error\n");
        exit(1);
    }
    pause();
    exit(0);
}
```

```
static void ssu_signal_handler1(int signo) {
    printf("Signal handler of SIGINT : %d\n", signo);
    printf("SIGQUIT signal is blocked : %d\n", signo);
    printf("sleeping 3 sec\n");
    sleep(3);
    printf("Signal handler of SIGINT ended\n");
}

static void ssu_signal_handler2(int signo) {
    printf("Signal handler of SIGQUIT : %d\n", signo);
    printf("SIGINT signal is blocked : %d\n", signo);
    printf("sleeping 3 sec\n");
    sleep(3);
    printf("Signal handler of SIGQUIT ended\n");
}
```

### 실행 결과

```
root@localhost:/home/oslab# ./ssu_sigaction_3
^CSignal handler of SIGINT : 2
SIGQUIT signal is blocked : 2
sleeping 3 sec
^CSignal handler of SIGINT ended
Signal handler of SIGINT : 2
SIGQUIT signal is blocked : 2
sleeping 3 sec
Signal handler of SIGINT ended
```