Signals and Signal Handling – Part 2

Signal Mask

- Be used to block signal delivery.
  - A blocked signal depends on the recipient process to unblock and handle it accordingly.
- A signal mask may be implemented using an integer.
  - Positional -- each bit corresponds to one signal.
  - Bit 1's -- the corresponding signals are being blocked.
- A process may query or change its signal mask by a call to sigprocmask( ).

sigprocmask (1)

- int sigprocmask(int *how*, const sigset_t **set*, sigset_t **oldset*);
  - Change the list of currently blocked signals.
  - Return 0 on success and -1 on error.
  - *oldset*
    - If non-null, the previous value of the signal mask is stored in *oldset*.
  - It is not possible to block SIGKILL or SIGSTOP with the sigprocmask call.
  - If *set* is NULL, *how* is ignored and the current value of the signal mask is returned by *oldset*.

sigprocmask (2)

- *how*
  - SIG_BLOCK
    - The set of blocked signals is the *union* of the current set and the *set* argument.
  - SIG_UNBLOCK
    - The signals in *set* are removed from the current set of blocked signals.
    - It is legal to attempt to unblock a signal which is not blocked.
  - SIG_SETMASK
    - The set of blocked signals is *set* to the argument *set*.

Example #7: sigprocmask (1)

```c
/* signal blocking -- sigprocmask의 사용예를 보인다. */
#include <unistd.h>
#include <signal.h>

int main() {
  sigset_t set1, set2;

  /* 시그널 집합을 완전히 채운다. */
  sigfillset (&set1);

  /* SIGINT와 SIGQUIT를 포함하지 않는 시그널 집합을 생성한다. */
  sigfillset (&set2);
  sigdelset (&set2, SIGINT);
  sigdelset (&set2, SIGQUIT);
  /* 중대하지 않은 코드를 수행 ... */
```

Example #7: sigprocmask (2)

```
 /* 봉쇄를 설정한다. */
  sigprocmask(SIG_SETMASK, &set1, NULL);

  /* 극도로 중대한 코드를 수행한다. */

  /* 하나의 봉쇄를 제거한다. */
  sigprocmask(SIG_UNBLOCK, &set2, NULL);


  /* 덜 중대한 코드를 수행한다 ... */


  /* 모든 시그널 봉쇄를 제거한다. */
  sigprocmask(SIG_UNBLOCK, &set1, NULL);
}
```

Example #8: sigismember

```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/types.h>

void pr_mask(const char *str)
{
  sigset_t sigset;

  if (sigprocmask(0, NULL, &sigset) < 0) {
    perror("sigprocmask error");
    exit(1);
  }

  printf("%s", str);

  if (sigismember(&sigset, SIGINT))
    printf("SIGINT");
  if (sigismember(&sigset, SIGQUIT))
    printf("SIGQUIT");
  if (sigismember(&sigset, SIGUSR1))
    printf("SIGUSR1");
  if (sigismember(&sigset, SIGALRM))
    printf("SIGALRM");

  /* remaining signals can go here */

  printf("\n");
}
```

sigpending

- #inclide <signal.h>

  int sigpending(sigset_t *set);
  - Examine the pending signals.
  - The signal mask of pending signals is stored in *set*.
  - Return 0 on success and -1 on error.

Example #9: sigpending (1)

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>

void fatal(const char *msg, int no){
  perror(msg);
  exit(no);
}
static void sig_quit(int signo) {
  printf("caught SIGQUIT\n");
  if (signal(SIGQUIT, SIG_DFL) == SIG_ERR)
    fatal("can't reset SIGQUIT", 1);
 return;
}
int main(void) {
  sigset_t newmask, oldmask, pendmask;

  if (signal(SIGQUIT, sig_quit) == SIG_ERR)
    fatal("can't catch SIGQUIT", 1);
```

Example #9: sigpending (2)

```c
  /* block SIGQUIT and save currnet signal mask */
  sigemptyset(&newmask);
  sigaddset(&newmask, SIGQUIT);
  if (sigprocmask(SIG_BLOCK, &newmask, &oldmask) < 0)
    fatal("SIG_BLOCK error", 1);
  sleep(5);

  if (sigpending(&pendmask) < 0)
    fatal("sigpending error", 1);
  if (sigismember(&pendmask, SIGQUIT))
    printf("\nSIGQUIT pending\n");

  /* reset signal mask which unblock SIGQUIT */
  if (sigprocmask(SIG_SETMASK, &oldmask, NULL) < 0)
    fatal("SIG_SETMASK error", 1);
  printf("SIGQUIT unblocked\n");
  /* SIGQUIT here will terminate with core file */
  sleep(5);
  exit(0);
}
```

```
> ./main
^\
SIGQUIT pending
caught SIGQUIT
SIGQUIT unblocked
^\Quit (core dumped)
> ./main
^\^\^\^\^\^\^\^\^\^\^\^\^\
SIGQUIT pending
caught SIGQUIT
SIGQUIT unblocked
^\Quit (core dumped)
```

Some Other System Calls

- kill( ), raise( )
  - kill( ) sends a signal to a process or a group of process.
  - raise( ) sends a signal to the calling process itself.
- alarm( )
  - Be used to set a timer that will expire at a specified time in the future.
- pause( )
  - Be used to suspend the calling process until a signal is received.
- sigsetjmp( ), siglongjmp( )

kill (1)

- #include <sys/types.h>

  #include <signal.h>

  int kill(pid_t *pid*, int *sig*);
  - Used to send any signal to any process group or process.

kill (2)

- *pid*
  - *pid* > 0: signal *sig* is sent to *pid* (*individual*).
  - 0: *sig* is sent to all processes that belong to the same process group of the sender.
  - -1: if the effective user-id of the process is superuser, then *sig* is sent to every process except for the system processes. If not, sig is sent to all processes with a real user-id equal to the effective user-id of the sender.
  - *pid* < -1: *sig* is sent to every process with a process group-id equal to the absulute value of *pid*
- *sig* is 0
  - No signal is sent, but error checking is still performed.
- Return value
  - On success, zero is returned.
  - On error, -1 is returned.

Example #10: kill (1)

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>

int ntimes = 0;

int main() {
    pid_t pid, ppid;
    void p_action (int), c_action (int);
    static struct sigaction pact, cact;

    /* 부모를 위해 SIGUSR1 행동을 지정한다. */
    pact.sa_handler = p_action;
    sigaction (SIGUSR1, &pact, NULL);

    switch (pid = fork()){
        case 1: /* 오류 */
        perror ("synchro");
        exit (1);
```

Example #10: kill (2)

```
        case 0: /* 자식 */
        /* 자식을 위해 행동을 지정 */
        cact.sa_handler = c_action;
        sigaction (SIGUSR1, &cact, NULL);
        ppid = getppid(); /* 부모의 프로세스 식별번호를 얻음. */
        for (;;) {
            sleep (1);
            kill (ppid, SIGUSR1);
            pause();
        } /* 결코 퇴장(exit) 않음. */

        default: /*  부모 */
        for(;;) {
            pause();
            sleep (1);
            kill (pid, SIGUSR1);
        } /* 결코 퇴장(exit) 않음 */
    }
}
```

Example #10: kill (3)

```c
void p_action (int sig) {
    printf ("Parent caught signal #%d\n", ++ntimes);
}

void c_action (int sig) {
    printf ("Child caught signal #%d\n", ++ntimes);
}
```

```
> ./main
Parent caught signal #1
Child caught signal #1
Parent caught signal #2
Child caught signal #2
Parent caught signal #3
Child caught signal #3
Parent caught signal #4
Child caught signal #4
Parent caught signal #5
Child caught signal #5
^Cexited, interrupt
```

raise

- #include <signal.h>

  int raise (int *sig*);
  - Sends a signal to the current process.
  - Semantically equals to kill(getpid( ), sig).
  - Return value
    - 0 on success, nonzero for failure.

alarm

- #include <unistd.h>

  unsigned int alarm(unsigned int *seconds*);
  - Arranges for a SIGALRM signal to be delivered to the process in *seconds* seconds.
  - If *seconds* is zero, no new alarm is scheduled
  - In any event any previously set alarm is cancelled.
  - Return value
    - The number of seconds remaining until any previously scheduled alarm was due to be delivered.
    - Zero if there was no previously scheduled alarm.

Example #11: alarm (1)

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

#define TIMEOUT 5 /* in seconds */
#define MAXTRIES 5
#define LINESIZE 1024
#define CTRL_G '\007' /* ASCII Bell */
#define TRUE 1
#define FALSE 0

/* used to see if timeout has occured */
static int timed_out;

/* will hold input line */
static char in_line[LINESIZE];

char *quickreply(char *prompt);
void catch(int signo);
```

Example #11: alarm (2)

```
int main(void) {
    char *name;
    if ((name = quickreply("Please enter your name")) != NULL)
        printf("Thank you, your name is %s\n", name);
    else
        printf("I give up!\n");
}

char *quickreply(char *prompt) {
    void (*was)(int);
    int ntries;
    char *answer;

    /* catch SIGALRM + save previous action */
    was = signal(SIGALRM, catch);

    for(ntries = 0; ntries < MAXTRIES; ntries++) {
        timed_out = FALSE;
        printf("\n%s > ", prompt);
```

Example #11: alarm (3)

```
        /* set alarm clock */
        alarm(TIMEOUT);

        /* get input line */
        answer = fgets(in_line, LINESIZE, stdin);

        /* turn off alarm */
        alarm(0);

        /* if timed_out TRUE, then no reply */
        if(!timed_out)
            break;
    }

    /* restore old action */
    signal(SIGALRM, was);

    /* return appropriate value */
    return (ntries == MAXTRIES ? ((char *) 0) : answer);
}
```

Example #11: alarm (4)

```c
/* executed when SIGALRM received */
void catch(int signo) {
    /* set timeout flag */
    timed_out = TRUE;

    /* ring a bell */
    putchar(CTRL_G);
    printf("bbb\n");
}
```

```
> ./main

Please enter your name > bbb


Please enter your name > hasoo
Thank you, your name is hasoo
```