Signals and Signal Handling – Part 3

pause

- #include <unistd.h>

  int pause(void);
  - Causes the invoking process to sleep until a signal is received.
  - Return value
    - Always returns -1.

Example #12: pause (1)

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <signal.h>
#include <unistd.h>
#define TRUE    1
#define FALSE   0
#define BELLS   "\007\007\007" /*ASCII bells */

int alarm_flag = FALSE;

/* SIGALRM을 처리할 루틴 */
void setflag (int sig) {
    alarm_flag = TRUE;
}


int main (int argc, char **argv) {
    int nsecs, j;
    pid_t pid;
    static struct sigaction act;
```

Example #12: pause (2)

```
if (argc<=2) { /* The tml means tell me later */
    fprintf(stderr, "Usage:  tml #seconds message\n");
    exit(1);
}

if ((nsecs = atoi(argv[1])) <= 0) {
    fprintf(stderr, "tml : invalid time\n");
    exit(2);
}

/* 백그라운드 프로세스를 생성하기 위해 fork한다. */
switch (pid = fork()){
    case -1: /* 오류 */
    perror("tml");
    exit(1);
    case 0: /* 자식 */
    break;
    default: /* 부모 */
    printf("tml processid %d\n", pid);
    exit(0);
}
```

Example #12: pause (3)

```
    /* 알람을 위한 행동을 지정한다. */
    act.sa_handler = setflag;
    sigaction(SIGALRM, &act, NULL);
    /* 알람 시계를 켠다. */
    alarm(nsecs);
    /* 시그널이 올 때까지 중단(pause) ... */
    pause();
    /* 만일 시그널이 SIGALRM이면 메시지를 프린트하라. */
    if (alarm_flag == TRUE) {
        printf (BELLS);
        for (j = 2; j < argc; j++)
            printf("%s ", argv[j]);
        printf("\n");
    }
    return 0;
}
```

```
ubuntu@server:~$ ./tml 10 happy day
tml process id 4988
ubuntu@server:~$ happy day
```

abort

- #include <stdlib.h>

  void abort(void);
  - Send the SIGABORT signal to the calling process, causing abnormal termination, i.e. a core dump.
  - If the abort( ) function causes program termination, all open streams are closed and flushed.
  - If the SIGABORT signal is ignored, then the ISO C implementation will still terminate the process, but other implementations may allow the abort( ) function to return to the caller.
  - Return value
    - The abort( ) function never returns.

sigsetjmp and siglongjmp

- #include <setjmp.h>

  int sigsetjmp(sigjmp_buf *env*, int *savemask*);

  void siglongjmp(sigjmp_buf *env*, int *val*);
  - Used for jumping from signal handler.
  - If *savemask* is non-zero, the current signal mask will be saved in *env* and restored when siglongjmp( ) is called from the signal handler.
  - Return value
    - sigsetjmp( ) returns 0 if called directly, non-zero (*val*) if returning from a call to siglongjmp( ).
    - siglongjmp( ) never return.

Example #13: siglongjmp (1)

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

static void sig_usr1(int), sig_alrm(int);
static sigjmp_buf jmpbuf;
static volatile sig_atomic_t canjump;

void pr_mask(const char *str) {
    sigset_t sigset;
    if (sigprocmask(0, NULL, &sigset) < 0) {
        perror("sigprocmask error");
        exit(1);
    }
    printf("%s", str);
```

Example #13: siglongjmp (2)

```
    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");
}

int main(void) {
    if (signal(SIGUSR1, sig_usr1) == SIG_ERR) {
        perror("signal (SIGUSR1) error");
        exit(1);
    }
    if (signal(SIGALRM, sig_alrm) == SIG_ERR) {
        perror("signal (SIGALRM) error");
        exit(1);
    }
```

Example #13: siglongjmp (3)

```
    pr_mask("starting main: ");
    if (sigsetjmp(jmpbuf, 1)) {
        pr_mask("ending main: ");
        exit(0);
    }
    canjump = 1; /* now sigsetjmp() is OK */
    for (;;) pause();
}

static void sig_usr1(int signo) {
    time_t starttime;
    if (canjump == 0)
        return; /* unexpected signal, ignore */

    pr_mask("starting sig_usr1: ");
    alarm(3); /* SIGALRM in 3 seconds */
    starttime = time(NULL);
    for (;;) /* busy wait for 5 seconds */
        if (time(NULL) > starttime + 5)
            break;
```

Example #13: siglongjmp (4)

```
    pr_mask("finishing sig_usr1: ");
    canjump = 0;
    /* jump back to main, don't return */
    siglongjmp(jmpbuf, 1);
}

static void sig_alrm(int signo) {
    pr_mask("in sig_alrm: ");
    return;
}
```

```
> ./main&
[1] 298
> starting main:
./kill -USR1 298
starting sig_usr1: SIGUSR1
> in sig_alrm: SIGUSR1SIGALRM
finishing sig_usr1: SIGUSR1
ending main:
^C
[1]+  Done                    ./main
```

More Examples #1: signal (1)

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

int pid;
static char *chdargv[] = {"cmd", "p1", (char *)0};
static char *chdenv[4];

int main (int argc, char *argv[]) {
    int i, cpid1, cpid2, cpid3;
    void wakeup(int), handler(int), trapper(int), parent(int);

    if (!(cpid1 = fork())) { /* 1st child */
        pid = cpid1 = getpid(); /* get pid for child 1 */
        printf("\nCPID1 = %d", cpid1);
```

More Examples #1: signal (2)

```
        // NSIG is definded in signal.h
        for (i = 1; i < NSIG; i++)
            signal(i, SIG_IGN); /* ignore all signals */

        signal(SIGINT, handler); /* except for interrupt */
        signal(SIGALRM, wakeup); /* and alarm clock */
        alarm((unsigned)2); /* set alarm for 2 secs */

        for (;;)
            pause(); /* wait for signals */

        printf(" -- CPID1 (%d) terminates\n", cpid1); /* never gets here */
        exit(0);
    } else {

        if (!(cpid2 = fork())) { /* 2nd child */
            pid = cpid2 = getpid(); /* get pid for child 2 */
            printf("\n\tCPID2 = %d", cpid2);
```

More Examples #1: signal (3)

```
        for (i = 1; i < NSIG; i++)
            signal(i, trapper); /* to trap signals */

        pause();
        printf(" -- CPID2 (%d) terminates\n", cpid2);
        exit(0);
    } else {
        if (!(cpid3 = fork())) { /* 3rd child */
            pid = cpid3 = getpid(); /* get pid for child 3 */
            printf("\n\t\tCPID3 = %d ", cpid3);
            signal(SIGUSR2, trapper);

            chdenv[0] = "HOME=here";
            chdenv[1] = "PATH=/bin";
            chdenv[2] = "MAILX=/usr/lib/xxx";
            chdenv[3] = (char *)0 ;

            pause();
```

More Examples #1: signal (4)

```
        printf(" -- CPID3 (%d) terminates\n", cpid3);
        exit(0);
}

/* parent process */
pid = getpid(); /* pid for parent */
sleep(3); /* to let child run first */
printf("\nThis is parent process (pid = %d)\n", pid);

for (i = 1; i < NSIG; i++)
        signal(i, parent); /* catch all signals */

printf("\n\tSend SIGBUS(%d) to CPID1 (%d)", SIGBUS, cpid1);
kill(cpid1, SIGBUS);

printf("\n\tSend SIGINT(%d) to CPID1 (%d)", SIGINT, cpid1);
kill(cpid1, SIGINT);

printf("\n\t\tSend SIGBUS(%d) to CPID2 (%d)", SIGBUS, cpid2);
kill(cpid2, SIGBUS);
```

More Examples #1: signal (5)

```
            printf("\n\t\tSend SIGUSR1(%d) to CPID2 (%d)", SIGUSR1, cpid2);
            kill(cpid2, SIGUSR1);

            printf("\n\t\t\tSend SIGUSR2(%d) to CPID3 (%d)", SIGUSR2, cpid3);
            kill(cpid3, SIGUSR2);

            waitpid(cpid3, NULL, 0);
        }
    }
    return 0;
} /* main */

void wakeup(int dummy) {
    printf("\nI (pid = %d) am up now\n", pid);
} /* wakeup */

void handler(int dummy) {
    printf("\nI (pid = %d) got an interrupt; will continue\n", pid);
} /* handler */
```

More Examples #1: signal (6)

```
void trapper(int i) {
    signal(i, trapper); /* old style; don't reset to default */

    if (i == SIGUSR1) {
        printf("\n\tGot SIGUSR1(%d); process(%d) will terminate\n", i, pid);
        exit(2);
    } else {
        if (i == SIGUSR2) {
            printf("\n\t\t(%d) got SIGUSR2(%d); execs a new program", i, pid);
            execve("./sigexec", chdargv, chdenv);
            perror("\nTHIS LINE SHOULDN'T BE HERE\n");
        } else
            printf("\n\tGot a signal(%d); process(%d) continues\n", i, pid);
    }
} /* trapper */

void parent(int sig) {
    printf("\nSignal (%d) received by parent (%d)\n", sig, pid);
} /* parent */
```

More Examples #1: signal (7)

```
// sigexec.c
#include <stdio.h>

extern char **environ;
int main(int argc, char * argv[]) {
    char **env;
    int i;

    printf("\nParameters are:\n");
    for (i=0; i<argc; i++)
        printf("%2d: %s\n", i, argv[i]);
    printf("\nEnviroment variable are:\n");
    for (env = environ; *env != (char *)0; env++)
        printf(" %s\n", *env);
    return 0;
}

# .replit
language = "c"
run = "clang-7 -pthread -lm -o main main.c;clang-7 -pthread -lm -o sigexec
sigexec.c;./main"
```

More Examples #1: signal (8)

```
CPID1 = 92
I (pid = 92) am up now

This is parent process (pid = 91)

    Send SIGBUS(7) to CPID1 (92)
    Send SIGINT(2) to CPID1 (92)

I (pid = 92) got an interrupt; will continue
        Send SIGBUS(7) to CPID2 (93)
        Send SIGUSR1(10) to CPID2 (93)
    CPID2 = 93
    Got SIGUSR1(10); process(93) will terminate
        CPID3 = 94
            Send SIGUSR2(12) to CPID3 (94)
Signal (17) received by parent (91)

Parameters are:
 0: cmd
 1: p1

Enviroment variable are:
 HOME=here
 PATH=/bin
 MAILX=/usr/lib/xxx

Signal (17) received by parent (91)
```

Thank you