실습 9주차

CPS LAB

시그널

목차

시그널

- 소프트웨어 인터럽트로 프로세스에 무엇인가 발생했음을 알리는 간단한 메시지를 비동기적으로 전달된 시그널은 시그널 핸들러에 의해 처리됨
  - 리눅스 운영체제가 프로세스에 전달한다
  - 예) 사용자가 Ctrl + C 같은 인터럽트 키를 입력한 경우

- 정의되어 있는 시그널: (signal.h 헤드파일에 정의됨)

```
kyhooon@kyh:~$ 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
kyhooon@kyh:~$
```

예제풀이

시그널 전달

Ctrl+c -> 2)SIGINT

Program Interrupt

```c
#include <stdio.h>

int main(void) {
        while(1);
        return 0;
}
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch8$ vim test.c
kyhooon@kyh:~/sysprogram_practice/p_ch8$ ./test.out
^C
kyhooon@kyh:~/sysprogram_practice/p_ch8$
```

Ctrl+z -> 20)SIGTSTP

Interactive Stop Signal :

Unlike SIGSTOP , this signal can be handled and ignored

```
kyhooon@kyh:~/sysprogram_practice/p_ch8$ ./test.out

[1]+  Stopped                 ./test.out
kyhooon@kyh:~/sysprogram_practice/p_ch8$
```

```
kyhooon@kyh:~$ ps -ef | grep test
kernoops     1366       1  0 10월 18 ?        00:02:44 /usr/sbin/kerneloops --test
kyhooon    765073  764925 99 05:16 pts/1    00:00:08 ./test.out
kyhooon    765077  765058  0 05:16 pts/2    00:00:00 grep --color=auto test
kyhooon@kyh:~$
kyhooon@kyh:~$ kill -20 765073
kyhooon@kyh:~$
```

예제풀이

시그널 전달

8)SIGFPE Floating Point Exception

```c
#include <stdio.h>

int main(void) {
        int a = 0;
        printf("%d\n", 3/a);
        while(1);
        return 0;
}
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch8$ vim test.c
kyhooon@kyh:~/sysprogram_practice/p_ch8$ ./test.out
Floating point exception (core dumped)
kyhooon@kyh:~/sysprogram_practice/p_ch8$
```

11)SIGSEGV Segmentation Fault

```c
#include <stdio.h>

int main(void) {

        char *str = "hello";

        *str = "world";

        return 0;
}
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch8$ vim test.c
kyhooon@kyh:~/sysprogram_practice/p_ch8$ gcc test.c -o test.out
test.c: In function 'main':
test.c:7:7: warning: assignment to 'char' from 'char *' makes in
out a cast [-Wint-conversion]
    7 |  *str = "world";
      |       ^
kyhooon@kyh:~/sysprogram_practice/p_ch8$ ./test.out
Segmentation fault (core dumped)
kyhooon@kyh:~/sysprogram_practice/p_ch8$
```

*메모리 불법 접근

시그널

- 프로그램에서 시그널을 보내려면 kill(), raise(), abort()함수를 사용함

```
#include <sys/types.h>
#include <signal.h>
```

The kill() system call can be used to send any signal to any process group or process

int kill(pid_t pid, int sig);

- pid, 시그널을 받을 프로세스의 PID
- sig, pid로 지정한 프로세스에 보내는 시그널

PID Value:

> 0, pid로 지정되어 있는 프로세스

< 0, 지정한 그룹에 시그널 전달함 (예, kill -9 -764925)

```
[kyhooon@kyh:~$ ps -ef | grep test
kernoops    1366       1  0 10월 18 ?      00:02:44 /usr/sbin/kerneloops --test
kyhooon   765073  764925  4 05:16 pts/1    00:01:33 ./test.out
kyhooon   765290  764925 99 05:48 pts/1    00:00:16 ./test.out
kyhooon   765362  765346  0 05:49 pts/2    00:00:00 grep --color=auto test
[kyhooon@kyh:~$ kill -9 -764925
[kyhooon@kyh:~$ ps -ef | grep test
kernoops    1366       1  0 10월 18 ?      00:02:44 /usr/sbin/kerneloops --test
kyhooon   765369  765346  0 05:49 pts/2    00:00:00 grep --color=auto test
```

0, 해당 프로세스의 그룹, 그 그룹에 있는 모든 프로세스 전달함

-1, 해당 프로세스 없음

예제풀이

예제 8-1

kill()함수로 시그널 보내기

- 무슨 뜻인지 찾도록 하자 (현재 Default 됨)
- man 7 signal

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

int main(void) {

        printf("Before SIGCONT Signal to parent.\n");
        kill(getppid(), SIGCONT);

        printf("Before SIGQUIT Signal to me.\n");
        kill(getpid(), SIGQUIT);

        printf("After SIGQUIT Signal.\n");

        return 0;
}
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch8$ ./ch8_1.out
Before SIGCONT Signal to parent.
Before SIGQUIT Signal to me.
Quit (core dumped)
kyhooon@kyh:~/sysprogram_practice/p_ch8$
```

Signal	Standard	Action	Comment
SIGABRT	P1990	Core	Abort signal from **abort**(3)
SIGALRM	P1990	Term	Timer signal from **alarm**(2)
SIGBUS	P2001	Core	Bus error (bad memory access)
SIGCHLD	P1990	Ign	Child stopped or terminated
SIGCLD	–	Ign	A synonym for **SIGCHLD**
SIGCONT	P1990	Cont	Continue if stopped
SIGEMT	–	Term	Emulator trap
SIGFPE	P1990	Core	Floating-point exception
SIGHUP	P1990	Term	Hangup detected on controlling terminal or death of controlling process
SIGILL	P1990	Core	Illegal Instruction
SIGINFO	–		A synonym for **SIGPWR**
SIGINT	P1990	Term	Interrupt from keyboard
SIGIO	–	Term	I/O now possible (4.2BSD)
SIGIOT	–	Core	IOT trap. A synonym for **SIGABRT**
SIGKILL	P1990	Term	Kill signal
SIGLOST	–	Term	File lock lost (unused)
SIGPIPE	P1990	Term	Broken pipe: write to pipe with no readers; see **pipe**(7)
SIGPOLL	P2001	Term	Pollable event (Sys V). Synonym for **SIGIO**
SIGPROF	P2001	Term	Profiling timer expired
SIGPWR	–	Term	Power failure (System V)
SIGQUIT	P1990	Core	Quit from keyboard
SIGSEGV	P1990	Core	Invalid memory reference
SIGSTKFLT	–	Term	Stack fault on coprocessor (unused)
SIGSTOP	P1990	Stop	Stop process
SIGTSTP	P1990	Stop	Stop typed at terminal
SIGSYS	P2001	Core	Bad system call (SVr4); see also **seccomp**(2)
SIGTERM	P1990	Term	Termination signal
SIGTRAP	P2001	Core	Trace/breakpoint trap
SIGTTIN	P1990	Stop	Terminal input for background process
SIGTTOU	P1990	Stop	Terminal output for background process
SIGUNUSED	–	Core	Synonymous with **SIGSYS**
SIGURG	P2001	Ign	Urgent condition on socket (4.2BSD)
SIGUSR1	P1990	Term	User-defined signal 1
SIGUSR2	P1990	Term	User-defined signal 2
SIGVTALRM	P2001	Term	Virtual alarm clock (4.2BSD)
SIGXCPU	P2001	Core	CPU time limit exceeded (4.2BSD); see **setrlimit**(2)
SIGXFSZ	P2001	Core	File size limit exceeded (4.2BSD); see **setrlimit**(2)
SIGWINCH	–	Ign	Window resize signal (4.3BSD, Sun)

시그널 핸들러 지정: signal(3)

```
#include <signal.h>

void (*signal(int sig, void (*disp)(int))(int);
```

- signal, 함수 포인터 이름
- sig, SIGINT/SIGQUIT/SIGKILL ...
  (즉, 시그널 핸들러로 처리하려는 시그널)

- void (*disp)(int), signal함수의 두번째 인자는, 또 하나의 함수 가르킨 포인터이다
  여기서 자신의 정의된 신호처리 함수를 해당
  - 예) void mySignalHandler(int sig) { /* 생략 */ }
    disp = mySignalHandler;

/* The behavior of signal() varies across UNIX versions, and has also varied historically across different versions of Linux. Avoid its use: use sigaction(2) instead. */

예제풀이

예제 8-2

signal()함수 사용하기

- 특정 신호에 대한 프로세스의 반응을 사용자가 정의할 수 있게 해줌

- 특정 신호를 받았을 때 운영 체제가 기본 동작을 수행하는 대신, 우리가 지정한 함수를 호출하도록 운영 체제에 지시할 수 있다

#include <signal.h>

psignal(int sig, const char *s);

- 함수는 두 번째 인자인 s로 지정한 문자열을 출력한 후 첫번째 인자인 sig로 지정한 시그널을 가리키는 이름을 붙여 표준 오류로 출력함

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

void sig_handler(int signo) {
        printf("Signal Handler signum: %d\n", signo);
        psignal(signo, "Received Signal");
}

int main(void) {

        void (*hand)(int);

        hand = signal(SIGINT, sig_handler);
        if( hand == SIG_ERR ) {
                perror("signal");
                exit(1);
        }


        printf("Wait 1st Ctrl+c ... : SIGINT\n");
        pause();
        printf("After 1st Signal Handler\n");
        printf("Wait 2nd Ctrl+c ... : SIGINT\n");
        pause();
        printf("After 2nd Signal Handler\n");

        return 0;
}
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch8$ ./ch8_2.out
Wait 1st Ctrl+c ... : SIGINT
^CSignal Handler signum: 2
Received Signal: Interrupt
After 1st Signal Handler
Wait 2nd Ctrl+c ... : SIGINT
^CSignal Handler signum: 2
Received Signal: Interrupt
After 2nd Signal Handler
kyhooon@kyh:~/sysprogram_practice/p_ch8$
```

시그널 핸들러 지정: sigset(3)

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

```
void (*sigset(int sig, void (*disp)(int))(int);
```

- signal, 함수 포인터 이름
- sig, SIGINT/SIGQUIT/SIGKILL …
  (즉, 시그널 핸들러로 처리하려는 시그널)

- void (*disp)(int), signal함수의 두번째 인자는, 또 하나의 함수 가르킨 포인터이다
  여기서 자신의 정의된 신호처리 함수를 해당
  - 예) void mySignalHandler(int sig) { /* 생략 */ }
    disp = mySignalHandler;

질문: signal 차이점?

- signal – POSIX 표준 | sigset – System V 유닉스
- ‘sigaction’ 함수가 이 두개 함수의 기능을 대체함

예제풀이

예제 8-4

sigset()함수로 시그널 핸들러 지정하기

_XOPEN_SOURCE_EXTENED

- C 프로그램을 컴파일할 때 특정 시스템 기능을 활성화하기 위해 사용되는 매크로
- UNIX 시스템 프로그래밍 인터페이스를 정의

```c
#define _XOPEN_SOURCE_EXTENDED 1
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

void sig_handler(int signo) {
        printf("Signal Handler Signal Number : %d\n", signo);
        psignal(signo, "Received Signal");
}

int main(void) {

        void (*hand)(int);

        hand = sigset(SIGINT, sig_handler);
        if( hand == SIG_ERR ) {
                perror("signal");
                exit(1);
        }

        printf("Wait 1st Ctrl+c ... : SIGINT \n");
        pause();
        printf("After 1st Signal Handler\n");
        printf("Wait 2st Ctrl+c ... : SIGINT \n");
        pause();
        printf("After 2st Signal Handler\n");

        return 0;
}
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch8$ ./ch8_4.out
Wait 1st Ctrl+c ... : SIGINT
^CSignal Handler Signal Number : 2
Received Signal: Interrupt
After 1st Signal Handler
Wait 2st Ctrl+c ... : SIGINT
^CSignal Handler Signal Number : 2
Received Signal: Interrupt
After 2st Signal Handler
kyhooon@kyh:~/sysprogram_practice/p_ch8$
```

시그널 집합 처리 함수들:

sigemptyset, sigfillset, sigaddset, sigdelset, sigismember -(3)

```
#include <signal.h>
int sigemptyset(sigset_t *set);                 // 시그널 집합 비우기
int sigfillset(sigset_t *set);                  // 시그널 집합에 모든 시그널 설정
int sigaddset(sigset_t *set, int signum);       // 시그널 집합에 시그널 설정 추가
int sigdelset(sigset_t *set, int signum);       // 시그널 집합에서 시그널 설정 삭제
int sigismember(sigset_t *set, int signum);     // 시그널 집합에 설정된 시그널 확인
```

[리눅스]

```
typedef struct {
        unsigned long __val[_NSIG_WORDS];
}sigset_t;
```

_NSIG_WORDS,
시스템 지원하는 신호의 총 수를 저장하는 워드의 개수
예) 64비트, 32비트

__val,
실제로 신호 집합의 비트 필드를 저장하는 배열이다
__val 배열 중, 각 비트 하나의 신호를 할당함

예제풀이

예제 8-5

시그널 집합 처리 함수 사용하기

__val 배열 직접 수정하는 것이 비추천!

- POSIX 표준 등 함수를 이용해야 함
  sigaddset, sigdelset, sigismember …

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

int main(void) {

        sigset_t st;

        sigemptyset(&st);

        sigaddset(&st, SIGINT);
        sigaddset(&st, SIGQUIT);

        if( sigismember(&st, SIGINT) )
                printf("SIGINT has been set. \n");

        printf("** Bit Pattern: %lx\n", st.__val[0]);

        return 0;
}
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch8$ vim ch8_5.c
kyhooon@kyh:~/sysprogram_practice/p_ch8$ ./ch8_5.out
SIGINT has been set.
** Bit Pattern: 6
kyhooon@kyh:~/sysprogram_practice/p_ch8$
```

sigaction() 함수의 활용:

signal, sigset 함수가 단순히 시그널 핸들러만 지정할 수 있는 것과 달리,
sigaction 함수는 훨씬 다양하게 시그널 제어할 수 있다

```
#include <signal.h>
int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);
signum,                    // 처리할 시그널
act,                       // 시그널 처리할 방법을 지정한 구조체 주소
oldact,                    // 기존에 시그널을 처리하던 방법을 저장할 구조체 주소

/* sigaction 함수는 signum에 지정한 시그널을 받았을 때, 처리할 방법을 sigaction 구조체인
   act로 받는다. */
```

[리눅스]

```
struct sigaction {
        int sa_flags;     // 0, sa_handler
                          // 1, sa_sigaction
        // sa_handler와 sa_sigaction 하나만 써야 한다 (배타적)
        void (*sa_handler)(int);
        void (*sa_sigaction)(int, siginfo_t *, void *);
        sigset_t sa_mask;
};
```

예제풀이

예제 8-6

sigaction 함수로 시그널 핸들러 지정하기

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

void sig_handler(int signo) {
        psignal(signo, "Received Signal: ");
        sleep(5);
        printf("In Signal Handler, After Sleep\n");
}

int main(void) {

        struct sigaction act;

        sigemptyset(&act.sa_mask);
        sigaddset(&act.sa_mask, SIGQUIT);
        act.sa_flags = 0;
        act.sa_handler = sig_handler;
        if( sigaction(SIGINT, &act, (struct sigaction *)NULL ) < 0 ) {
                perror("sigaction");
                exit(1);
        }

        fprintf(stderr, "Input SIGINT: ");
        pause();
        fprintf(stderr, "After Signal Handler\n");

        return 0;
}
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch8$ ./ch8_6.out
Input SIGINT: ^CReceived Signal: : Interrupt
^CIn Signal Handler, After Sleep
Received Signal: : Interrupt
In Signal Handler, After Sleep
After Signal Handler
kyhooon@kyh:~/sysprogram_practice/p_ch8$
```

예제 8-8

시그널 발생 원인 검색하기(SA_SIGINFO 설정)

sys/siginfo.h 헤드파일에 정의되어 있음

```
#include <siginfo.h>
void psiginfo(const siginfo_t *pinfo,
                        const char *s);
```

pinfo, 시그널 발생원인에 관한 정보(구조체 포인터)

s, 출력할 문자열

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

void sig_handler(int signo, siginfo_t *sf, ucontext_t *uc) {
        psiginfo(sf, "Received Signal: ");
        printf("si_code: %d\n", sf->si_code);
}

int main(void) {

        struct sigaction act;

        act.sa_flags = SA_SIGINFO;

        act.sa_sigaction = (void (*)(int, siginfo_t *, void *))sig_handler;
        sigemptyset(&act.sa_mask);

        if( sigaction(SIGUSR1, &act, (struct sigaction *)NULL) < 0 ) {
                perror("sigaction");
                perror("sigaction");
                exit(1);
        }

        pause();

        return 0;
}
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch8$ ./ch8_8.out &
[1] 736006
kyhooon@kyh:~/sysprogram_practice/p_ch8$ kill -USR1 736006
Received Signal: : User defined signal 1 (Signal sent by kill() 735178 1000)
si_code: 0
kyhooon@kyh:~/sysprogram_practice/p_ch8$
```

알람 시그널 함수들:

```
#include <unistd.h>
// 알람 시그널 생성
unsigned int alarm(unsigned int seconds);

// 타이머 정보 검색
int getitimer(int which, struct itimerval *curr_value);
```

which, 검색할 타이머의 종류

curr_value, 타이머 정보를 저장할 구조체 포인터

```
// 타이머 설정
int setitimer(int which, const struct itimerval *value, struct itimerval *ovalue);
```

value, 설정할 타이머의 정보를 저장한 구조체 포인터

ovalue, 이전 타이머의 정보를 저장할 구조체 포인터

[리눅스]

```
struct itimerval {
        struct timeval it_interval;
        struct timeval it_val;
};

struct timeval {
        time_t          tv_sec;
        suseconds_t tv_usec;
};
```

예제 8-10

인터벌 타이머 설정하기

- 실제 시간 기준 2초 간격으로 타이머가 작동하도록 만듬

타이머의 종류:

ITIMER_REAL:

실제 시간을 사용한다

ITIMER_VIRTUAL:

프로세스가 사용하는 사용자 모드 CPU 시간을 사용

ITIMER_PROF:

프로세스가 사용하는 시스템 모드 CPU 시간을 사용

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

void sig_handler() {
        printf("Timer Invoked .. \n");
}

int main(void) {

        struct itimerval it;

        signal(SIGALRM, sig_handler);
        it.it_value.tv_sec = 3;
        it.it_value.tv_usec = 0;
        it.it_interval.tv_sec = 2;
        it.it_interval.tv_usec = 0;

        if( setitimer(ITIMER_REAL, &it, (struct itimerval *)NULL) == -1 ) {
                perror("setitimer");
                exit(1);
        }

        while(1) {

                if( getitimer(ITIMER_REAL, &it) == -1 ) {
                        perror("getitimer");
                        exit(1);
                }

                printf("%d sec, %d msec.\n", (int)it.it_value.tv_sec, (int)it.it_value.tv_usec);
                sleep(1);
        }

        return 0;
}
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch8$ ./ch8_10.out
2 sec, 999995 msec.
1 sec, 999611 msec.
0 sec, 999460 msec.
Timer Invoked ..
1 sec, 999880 msec.
0 sec, 999744 msec.
Timer Invoked ..
1 sec, 999888 msec.
0 sec, 999750 msec.
Timer Invoked ..
1 sec, 999887 msec.
0 sec, 999749 msec.
Timer Invoked ..
1 sec, 999892 msec.
0 sec, 999754 msec.
^C
kyhooon@kyh:~/sysprogram_practice/p_ch8$
```

기타 시그널 함수들:

예제 8-11

sighold()함수로 시그널 블로킹하기

```
#include <signal.h>
// 시그널 정보출력
void psignal(int sig, const char *s);

// 시그널 정보 출력 (#include <string.h>)
char* strsignal(int sig);

// 시그널 블로킹과 해제
int sighold(int sig);
Int sigrelse(int sig);
```

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

void sig_handler(int signo) {
        char *s;

        s = strsignal(signo);
        printf("Received Signal : %s\n", s);
}

int main(void) {

        if( signal(SIGINT, sig_handler) == SIG_ERR ) {
                perror("signal");
                exit(1);
        }

        sighold(SIGINT);

        pause();

        return 0;
}
```

연습 문제

감사합니다.

CPS LAB

Signal Set

참조