9주차 2차시 시그널 기본 처리

【학습목표】

1. 시그널을 받아서 처리하는 방법을 설명할 수 있다.
2. 시그널 집합의 개념을 설명할 수 있다.

학습내용1 : 시그널 핸들러 함수

1. 시그널 보내기 : kill 명령

* 개념
- 프로세스에 시그널을 보내는 명령
- 예 : 3255번 프로세스에 9번 시그널(SIGKILL) 보내기 -> 프로세스 강제 종료

```
# kill -9 3255
```

* 함수 원형
pid에 대응하는 프로세스에 sig로 지정한 시그널 보냄
sig 0(NULL 시그널) : 실제로 시그널 보내지 않고 정상인지 오류인지 확인

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

int kill(pid_t pid, int sig);
```

pid : 시그널을 받을 프로세스의 ID
sig : pid로 지정한 프로세스에 보내려는 시그널

* pid 값에 따른 처리
① pid가 0보단 큰 수 : pid로 지정한 프로세스에 시그널 발송
② pid가 -1이 아닌 음수 : 프로세스 그룹ID가 pid의 절대값인 프로세스 그룹에 속하고 시그널을 보낼 권한을 가지고 있는 모든 프로세스에 시그널 발송
③ pid가 0 : 특별한 프로세스를 제외하고 프로세스 그룹ID가 시그널을 보내는 프로세스의 프로세스 그룹ID와 같은 모든 프로세스에게 시그널 발송
④ pid가 -1 : 시그널을 보낸는 프로세스의 유효 사용자ID가 root가 아니면, 특별한 프로세스를 제외하고 프로세스의 실제 사용자ID가 시그널을 보내는 프로세스의 유효 사용자ID와 같은 모든 프로세스에 시그널 발송

* kill 함수 사용하기

```
01  #include <sys/types.h>
02  #include <unistd.h>
03  #include <signal.h>
04  #include <stdio.h>
05
06  int main(void) {
07      printf("Before SIGCONT Signal to parent.\n");
08
09      kill(getppid(), SIGCONT);
10
11      printf("Before SIGQUIT Signal to me.\n");
12
13      kill(getpid(), SIGQUIT);
14
15      printf("After SIGQUIT Signal.\n");
16
17      return 0;
18  }
```

SIGQUIT의 기본동작은 코어덤프

```
# ex7_1.out
Before SIGCONT Signal to parent.
Before SIGQUIT Signal to me.
끝(Quit)(코어 덤프)
```

2. 시그널 보내기: raise(2)

* 함수를 호출한 프로세스에 시그널 발송

```
#include <signal.h>

int raise(int sig);
```

sig : 보내려는 시그널 번호

3. 시그널 보내기: abort(3)

* 함수를 호출한 프로세스에 SIGABRT시그널 발송
SIGABRT 시그널은 프로세스를 비정상적으로 종료시키고 코어덤프 생성

```
#include <stdlib.h>

void abort(void);
```

프로세스 종료. 리턴하지 않는다.

4. 시그널 핸들러 지정: signal(3)

* 함수 원형

```
#include <signal.h>

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

sig : 시그널 핸들러로 처리하려는 시그널
disp : 시그널 핸들러 함수명

* disp 인자 - 다음 세가지중 하나 설정
SIG_IGN : 시그널을 무시하도록 지정
SIG_DFL : 기본 처리 방법으로 처리하도록 지정
signal함수는 시그널이 들어올 때마다 시그널 핸들러를 호출하려면 매번 시그널 핸들러를 재지정해야함.

* signal 함수 사용하기

```
...
07  void handler(int signo) {
08      printf("Signal Handler Signal Number : %d\n", signo);
09      psignal(signo, "Received Signal");
10  }
11
12  int main(void) {
13      void (*hand)(int);
14
15      hand = signal(SIGINT, handler);
16      if (hand == SIG_ERR) {
17          perror("signal");
18          exit(1);
19      }
20
21      printf("Wait 1st Ctrl+C... : SIGINT\n");
22      pause();
23      printf("After 1st Signal Handler\n");
24      printf("Wait 2nd Ctrl+C... : SIGINT\n");
25      pause();
26      printf("After 2nd Signal Handler\n");27
28      return 0;
29  }
```

```
# ex7_2.out
Wait 1st Ctrl+C... : SIGINT
^CSignal Handler Signal Number : 2
Received Signal: Interrupt
After 1st Signal Handler
Wait 2nd Ctrl+C... : SIGINT
^C#
```

두번째 Ctrl+C는 처리못함

5. 시그널 핸들러 지정: sigset(3)

* sigset()
sigset함수는 signal함수와 달리 시그널 핸들러가 한 번 호출된 후에 기본동작으로 재설정하지 않고, 시그널 핸들러를 자동으로 재지정한다.
성공 시 시그널 핸들러 함수의 주소
실패 시 SIG_ERR 리턴

```
#include <signal.h>

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

sig : 시그널 핸들러로 처리하려는 시그널
disp : 시그널 핸들러 함수명

* sig로 지정한 시그널을 받았을 때 처리할 방법
SIG_IGN : 시그널을 무시하도록 지정
SIG_DFL : 기본 처리 방법으로 처리하도록 지정

* sigaction 구조체

```
...
07  void handler(int signo) {
08      printf("Signal Handler Signal Number : %d\n", signo);
09      psignal(signo, "Received Signal");
10  }
11
12  int main(void) {
13      if (sigset(SIGINT, handler) == SIG_ERR) {
14          perror("sigset");
15          exit(1);
16      }
17
18      printf("Wait 1st Ctrl+C... : SIGINT\n");
19      pause();
20  printf("After 1st Signal Handler\n");
21  printf("Wait 2nd Ctrl+C... : SIGINT\n");
22      pause();
23  printf("After 2nd Signal Handler\n");
24
25      return 0;
26  }
```

시그널 핸들러를 재지정 하지 않아도됨

```
# ex7_4.out
Wait 1st Ctrl+C... : SIGINT
^CSignal Handler Signal Number: 2
Received Signal: Interrupt
After 1st Signal Handler
Wait 2nd Ctrl+C... : SIGINT
^CSignal Handler Signal Number: 2
Received Signal: Interrupt
After 2nd Signal Handler
```

학습내용2 : 시그널 집합(signal set)

1. 시그널 집합의 개념

* 시그널 집합이란?
시그널을 개별적으로 처리하지 않고 복수의 시그널을 처리하기 위해 도입한 개념
시그널 : 비트 마스트(bit mask)로 표현, 각 비트가 특정 시그널과 1:1로 연결
비트 값이 1이면 해당 시그널 설정
비트 값이 0이면 시그널 설정 않음.

* 시그널 집합의 처리를 위한 구조체 : sigset_t

```
typedef struct {
        unsigned int __sigbits[4];
} sigset_t;
```

2. 시그널 집합 비우기 : sigemptyset(3)

* 시그널 집합에서 모든 시그널을 0으로 설정

```
#include <signal.h>

int sigemptyset(sigset_t *set);
```

set : 비우려는 시그널 집합의 주소

3. 시그널 집합에 모든 시그널 설정: sigfillset(3)

* 시그널 집합에서 모든 시그널을 1로 설정

```
#include <signal.h>

int sigfillset(sigset_t *set);
```

set : 설정하려는 시그널 집합의 주소

4. 시그널 집합에 시그널 설정 추가: sigaddset(3)

signo로 지정한 시그널을 시그널 집합에 추가

```
#include <signal.h>

int sigaddset(sigset_t *set, int signo);
```

set : 시그널을 추가하려는 시그널 집합의 주소
segno : 시그널 집합에 추가로 설정하려는 시그널

5. 시그널 집합에서 시그널 설정 삭제: sigdelset(3)

signo로 지정한 시그널을 시그널 집합에서 삭제

```
#include <signal.h>

int sigdelset(sigset_t *set, int signo);
```

set : 시그널을 삭제하려는 시그널 집합의 주소
segno : 시그널 집합에서 삭제하려는 시그널

6. 시그널 집합에 설정된 시그널 확인: sigismember(3)

signo로 지정한 시그널이 시그널 집합에 포함되어 있는지 확인

```
#include <signal.h>

int sigismember(sigset_t *set, int signo);
```

set : 확인하려는 시그널 집합의 주소
segno : 시그널 집합에서 설정되었는지 확인하려는 시그널

* 시그널 집합 처리 함수 사용하기

```
01  #include <signal.h>
02  #include <stdio.h>
03
04  int main(void) {
05      sigset_t st;
06
07      sigemptyset(&st);
08
09      sigaddset(&st, SIGINT);
10      sigaddset(&st, SIGQUIT);
11
12      if (sigismember(&st, SIGINT))
13          printf("SIGINT is setting.\n");
14
15      printf("** Bit Pattern: %x\n",st.__sigbits[0]);
16
17      return 0;
18  }
```

시그널 집합 비우기

시그널 추가

시그널 설정 확인

6은 2진수로 00000110이므로 오른쪽에서 2번, 3번 비트가 1로 설정
SIGINT는 2번, SIGQUIT는 3번 시그널

```
# ex7_5.out
SIGINT is setting.
** Bit Pattern: 6
```

【학습정리】

1.

① 시그널 핸들러 함수

기능	함수원형

시그널 보내기	int kill(pid_t pid, int sig);
	int raise(int sig);
	void abort(void);
	int sigsend(idtype_t idtype, id_t id, itn sig);

기능	함수원형

시그널 핸들러 지정	void (*signal (int sig, void (*disp)(int)))(int);
	void (*sigset(int sig, void (*disp)(int)))(int);
	int sigignore(int sig);

2. 시그널 집합(signal set)

* 시그널 집합이란?

- 시그널을 개별적으로 처리하지 않고 복수의 시그널을 처리하기 위해 도입한 개념
- 시그널

: 비트 마스트(bit mask)로 표현, 각 비트가 특정 시그널과 1:1로 연결

: 비트 값이 1이면 해당 시그널 설정

: 비트 값이 0이면 시그널 설정 않음.

- 시그널 집합의 처리를 위한 구조체 : sigset_t

```
typedef struct {
        unsigned int __sigbits[4];
} sigset_t;
```

3.

기능	함수원형

시그널 집합	int sigemptyset(sigset_t *set);
	int sigfillset(sigset_t *set);
	int sigaddset(sigset_t *set, int signo);
	int sigdelset(sigset_t *set, int signo);
	int sigsmember(sigset_t *set, int signo);