HALLYM UNIVERSITY

CIC Convergence Information & Communications

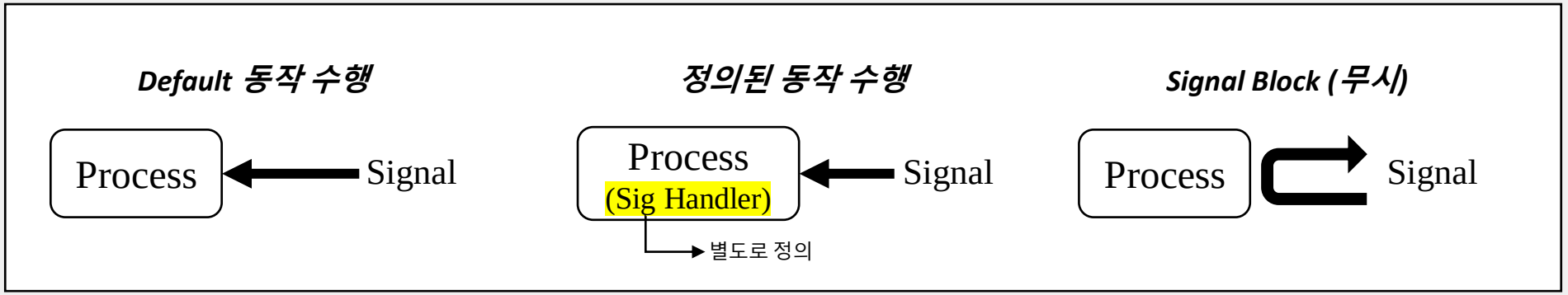
(Operating System) Practice -8-

Signal 1

Index

Signal Overview

- **Signal**
  - 비동기적 사건 (이벤트, 예외 등)의 발생을 프로세스에게 알리기 위해 사용
  - 다양한 종류의 Signal이 존재
    - ✓ 대부분의 Signal은 프로세스를 종료시키기 위해 사용됨
  - Signal을 수신한 프로세스는 다음 중 하나의 동작을 수행
    - ① 수신한 Signal의 Default 동작을 수행
    - ② 프로세스에 정의된 동작 수행
    - ③ Signal Block (무시)



Signal Practice -1 (Signal Handler)

- **Signal Handler**
  - ✓ **Signal** 함수는 Signal을 수신하면 어떻게 처리할지 정의함

유형	설명
헤더 파일	<signal.h>
형태	signal(int signum, void handler(int sig))
인수	int signum : signal 번호 void handler(int sig) : signal을 처리할 handler
반환 값	void *()(int) : 이전에 설정된 signal handler

  - ✓ Handler의 유형

유형	설명
함수 이름	signal을 받으면 해당 함수가 실행됨
SIG_IGN	signal을 받으면 무시함
SIG_DFL	signal을 받으면 시스템에 기본적으로 설정된 동작을 실행함

Signal practice -1 (Signal Handler)

- **Signal Handler Practice**

Signal1.c

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

void interruptHandler (int sig){
    printf("This process will be exited in 3 seconds\n");
    sleep(3);
    exit(0);
}
void main(){
    while(1){
        signal(SIGINT, interruptHandler);
        printf("Input Ctrl+c\n");
    }
}
```

Signal2.c

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

void stopHandler (int sig){
    printf("This process will be exited in 3 seconds\n");
    sleep(3);
    exit(0);
}
void main(){
    while(1){
        signal(SIGSTOP, stopHandler);
        printf("Input Ctrl+Z\n");
    }
}
```

SIGKILL과 SIGSTOP은 제어할 수 없다.

Signal Practice -2 (Pause)

- **Pause**
  - ✓ Pause 함수를 호출한 프로세스는 임의의 Signal을 수신할 때 까지 일시정지

유형	설명
헤더 파일	<unistd.h>
형태	int pause(void);
인수	void
반환 값	항상 : -1 errno : ERESTARTNOHAND로 설정됨

Pause.c

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

void inter_handler(int sig){
        printf("You pressed Ctrl+C\n");
}

int main(void){
        printf("Waiting for Ctrl+C\n");
        signal(SIGINT, inter_handler);
        pause();
        printf("End of pause\n");
}
```

Signal Practice -3 (Kill)

- **Kill**
  - ✓ **Kill** 함수는 프로세스에 Signal을 전송할 때 사용
  - ✓ Shell에서 사용되는 kill 명령어와는 다름

유형	설명
헤더 파일	<signal.h>
형태	int kill(pid_t pid, int sig);
인수	pid_t pid : signal을 받을 프로세스의 ID int sig : 전송할 signal 번호
반환 값	성공 : 0 실패 : -1

  - ✓ pid 인자 값 의미

pid	설명
양의 정수	지정한 프로세스 ID에만 signal 전송
0	함수를 호출하는 프로세스와 같은 그룹에 있는 모든 프로세스에 signal 전송
-1	함수를 호출하는 프로세스가 전송할 수 있는 권한을 가진 모든 프로세스에 signal 전송
-1 이외의 음수	첫번째 인수 pid의 절대값 프로세스 그룹에 속하는 모든 프로세스에 signal 전송

Signal Practice -3 (Kill)

- **Kill**

Kill1.c

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

int main(int argc, char **argv[]){
    while(1){
        printf("running\n");
        sleep(5);
    }
}
```

Kill2.c

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

int main(int argc, char *argv[]){
    kill(atoi(argv[1]), SIGKILL);
}
```

아래와 같이 실행

```
ubuntu-pc@ubuntupc-VirtualBox:~/practice8$ ./Kill1 &
[5] 9974
ubuntu-pc@ubuntupc-VirtualBox:~/practice8$ running
running
./Kill2 9974
[5]   Killed                  ./Kill1
ubuntu-pc@ubuntupc-VirtualBox:~/practice8$
```

Signal Practice -4 (Raise & Alarm)

- **Raise & Alarm**
  - ✓ **Raise** 함수와 **Alarm** 함수는 프로세스가 자기 자신에게 Signal을 전송하기 위해 사용
  - ✓ **Raise** 함수: 프로세스가 자기 자신에게 Signal 전송

유형	설명
헤더 파일	<signal.h>
형태	int raise(int sig);
인수	sig : 전송하려는 signal 번호
반환 값	성공 : 0 실패 : 0 이외의 값

  - ✓ **Alarm** 함수: 특정시간이 지나면, 자기 자신에게 SIGALRM Signal 전송
    - SIGALRM의 기본 동작은 프로세스의 종료

유형	설명
헤더 파일	<unistd.h>
형태	unsigned alarm(unsigned int seconds);
인수	seconds : 초 후에 signal 전송
반환 값	이전에 호출한 alarm 없을 경우 : 0 이전에 호출한 alarm 있을 경우 : 남은 시간

Signal Practice -4 (Raise & Alarm)

- **Raise & Alarm**

raise.c

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

int main(int argc, char **argv){
    int count =0;
    while(1){
        printf("Hello\n");
        count++;
        if(count==3){
            raise(SIGINT);
        }
        sleep(2);
    }
}
```

alarm.c

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

void alarmHandler(){
    printf("ding dong\n");
}

int main(int args, char **argv){
    signal(SIGALRM, alarmHandler);
    while(1){
        alarm(2);
        printf("waiting for alarm\n");
        pause();
    }
}
```

부록

- Signal의 종류

Signal	Portable number	Default Action	Description
SIGABRT	6	Terminate (core dump)	Process abort signal.
SIGALRM	14	Terminate	Alarm clock.
SIGBUS	n/a	Terminate (core dump)	Access to an undefined portion of a memory object.
SIGCHLD	n/a	Ignore	Child process terminated, stopped, or continued.
SIGCONT	n/a	Continue	Continue executing, if stopped.
SIGFPE	n/a	Terminate (core dump)	Erroneous arithmetic operation.
SIGHUP	1	Terminate	Hangup.
SIGILL	n/a	Terminate (core dump)	Illegal instruction.
SIGINT	2	Terminate	Terminal interrupt signal.
SIGKILL	9	Terminate	Kill (cannot be caught or ignored).
SIGPIPE	n/a	Terminate	Write on a pipe with no one to read it.
SIGPOLL	n/a	Terminate	Pollable event.
SIGPROF	n/a	Terminate	Profiling timer expired.
SIGQUIT	3	Terminate	Terminal quit signal.

부록

- Signal의 종류

Signal	Portable number	Default Action	Description
SIGSEGV	n/a	Terminate (core dump)	Invalid memory reference.
SIGSTOP	n/a	Stop	Stop executing (cannot be caught or ignored).
SIGSYS	n/a	Terminate (core dump)	Bad system call.
SIGTERM	15	Terminate	Termination signal.
SIGTRAP	n/a	Terminate (core dump)	Trace/breakpoint trap.
SIGTSTP	n/a	Stop	Terminal stop signal.
SIGTTIN	n/a	Stop	Background process attempting read.
SIGTTOU	n/a	Stop	Background process attempting write.
SIGUSR1	n/a	Terminate	User-defined signal 1.
SIGUSR2	n/a	Terminate	User-defined signal 2.
SIGURG	n/a	Ignore	High bandwidth data is available at a socket.
SIGVTALRM	n/a	Terminate	Virtual timer expired.
SIGXCPU	n/a	Terminate (core dump)	CPU time limit exceeded.
SIGXFSZ	n/a	Terminate (core dump)	File size limit exceeded.