안정적인 신호 인터페이스

X/Open과 Unix 명세는 signal함수 보다는 좀 더 안정적인, sigaction 이라는 새로운 함수를 권장한다.

Int sigaction(int sig , const struct sigaction \*act , struct sigaction \*oact);

Struct sigaction{

Void(\*) (int) sa\_handler ;

Sigset\_t sa\_mask ;

Int sa\_flags;

}

Sigaction 함수는 sig에 해당하는 신호에 연관된 행동을 설정한다. Oact가 NULL이 아니라면 sigaction은 oact가 가리키는 구조체에 이전의 신호 행동을 기록한다. Act가 Null이면 sigaction은 더 이상의 일을 하지 않는다. 반면 Null이 아니면 sig로 지정된 신호에 대해 act가 가리키는 행동을 설정한다.

SIG\_IGN과 SIG\_DFL을 sa\_handler에 설정할 수도 있다.

Sa\_mask는 sa\_handler함수가 호출되기 전에 프로세스의 신호 마스크에 추가될 신호들의 집합을 지정한다. 이 집합의 신호들은 차단된다. 예제를 통해서 확실히 알아볼 것!!

Sigaction에 의해 설정된 신호 처리부가 자동으로 해제되는 것은 아니다. 따라서 자동으로 해제되도록 하기 위해서는 sa\_flags 멤버에 반드시 SA\_RESETHAND를 추가해야 한다.

신호 집합

#include <signal.h>

Int sigaddset(sigset\_t \*set , int signo)

Int sigemptyset(sigset\_t \*set);

Int sigfillset(sigset\_t \*set);

Int sigdelset(sigset\_t \*set , int signo);

#include <signal.h>

Int sigismember(sigset\_t \*set , int signo)

신호집합에 해당 신호가 들어있으면 1 아니면 -1

#include <signal.h>

int sigprocmask(int how , const sigset\_t \*set , sigset\_t \*oset);

how : SIG\_BLOCK , SIG\_SETMASK , SIG\_UNBLOCK

set이 NULL 포인터이면 how의 값은 사용되지 않는다. 이런 경우 함수 호출의 유일한 목적은 현재 신호 마스크 값을 oset에 기록하는 것이다.

Sigaction 플래그들 (sa\_flags) ppt 보고 정리 및 예제 소스코드 확인하기