**[ Assignment 4 ]**

**수금반 Team 27**

**컴퓨터전공 2013011253 김기원**

**컴퓨터전공 2014004248 강준규**

**Signal**

**Notice : 코드의 LBL(Line By Line)은 소스코드가 있어, 생략하며 중요한 부분만 설명하였습니다.**

1. **stop.c**

1번 과제에서 제일 중요한 점이라면, sigaction 함수의 사용입니다. 우리가 Ctrl+C 로 주는 Signal 의 분류는 SIGINT(Interrupt)이기 때문에 간단한 sigaction(SIGINT, &mysig, 0) 과 함께, mysig.sa\_handler = sig\_fn 으로 정의하면 됩니다. 이는 SIGINT(Ctrl + C) 의 signal 이 들어올 때 mysig 라는 핸들러를 작동하며, 해당 핸들러는 sig\_fn 함수가 정의된 함수포인터다. 포인터를 통해 sig\_fn을 실행시키는 것이다. Sig\_fn 안에는 간단한 printf 면된다.

1. **count.c**

2번 과제는 난이도가 조금 있으며, 예전에 배웠던 File I/O 와 Process 와 Enviroment 의 Time 까지 Review 해주는 매우 좋은 과제입니다.

본 과제에서의 해결 난제 중 하나는 signal 함수 사용시 해당 함수는 parameter를 전달하는 방법이 없다는 것이다.

Signal 과제인 만큼 File I/O + Process fork + Time 부분은 생략하겠습니다.(In SouceCode)

Process fork 작업 이후, 프로세스들은 각각 pid1 , pid2 , pid3 를통해 자신의 pid 를 가지고 있습니다. 이후 주 목적인 While 문에 들어가, cpid = getpid() 를 통해 pid 를 check 하여, 지정된 프로세스만 들어갈 수 있는 if else 문들로 구성되어 있습니다.

if((cpid = getpid()) == pid1)

{

while(1)

{

signal(SIGUSR1,myfunc);

count = CnW(fd, filecount, count);

if(deadline == count)

{

gettimeofday(&endT,NULL);

Timeuse = 1000000\*(endT.tv\_sec - startT.tv\_sec) + (endT.tv\_usec - startT.tv\_usec);

Timeuse /= 1000000;

printf("Program Run Time : %f sec\n",Timeuse);

return 0;

}

kill(pid, SIGUSR1);

pause();

}

}

조건문을 진입한 프로세스마다 무한루프 while(1) 이 기다리고 있어, 안에는 signal 발생시 처리하는 명령들로 구성되어 있습니다. 명령을 수행(count++ , write처리)을 마친 프로세스는 깊은 잠으로 들어가기 전에, Kill(pid, SIGUSR1) 을 통해 자신의 자식 프로세스에게 signal 을 보내도록 합니다. 마지막 자식 프로세스는 자신이 끝난 후, 첫 프로세스를 깨워주는 작업이 필요합니다. 여기서 중요한 것이, 프로세스는 fork 이후 자신에 해당되는 if else문을 거르다 보면, getpid, getppid 로 자신 혹은 부모 의 pid만 볼수 있습니다. 하지만 두번째 fork 작업이 진행된 후, if else 문을 진입 전 pid1 = getppid(); [현재 pid2 상황에서] 를 해주어 지속적으로 pid1 을 유지하게 해준다면, 손자 프로세스도 할아버지의 PID 를 가지고 있게 됩니다.

SIGUSR1 은 myfunc 을 action 하게 되어 있지만, 사실상 read/write, count++ 를 수행하여야 하기 때문에 단순한 void 형 함수를 달기엔 무리가 있었습니다.

void myfunc(int signo){}

myfunc 은 signal 함수가 terminal 로 반환되지 않도록 빈 함수를 만들어 줬습니다.

Argv 인자로 받은 argv[1] 문자열은 atoi 를 통해 deadline 이라는 변수에 값을 저장해, count 값과 비교하여 더 이상 count 가 필요 없으면 즉시 return 하도록 사용하였습니다. Signal 을 Sleep 하는 함수로는 pause 를 사용한 이유도, signal이 도착하면 kill 에 해당하는 프로세스에 대해 pause가 풀리는 것 때문에 사용하였습니다.

(write 작업을 수행할 때 , 10만번을 수행하기 위한 버퍼 사이즈는 조절이 가능합니다.)

char filecount[1000000]; //Can Minimize or Maximize

1. **mysignaltest.c**

3번 과제는 코드 분석 능력과 함께 깊이 있는 공부를 유도하였습니다.

겉으로는 setjmp , longjmp 로 인한 가벼운 오류로 여겨지지만, 실제로는 signal 의 원초적인 원리 알아야 해결할 수 있는 문제입니다.

signal 함수는 해당 signal 이 발생하면 Action 을 취할 함수를 불러 실행시키지만, Action 을 취하는 중 만약 다른 signal 이 온다면, 구(舊) 시그널이 손실되는 경우가 생기는 것이 답안입니다. 해당 문제로, 중간에 SIGALRM 이 들어와 SIGINT 를 손실하는 경우입니다.

그것이 바로 왜 5초가 지나면 sleep2 만 실행하는 이유입니다.

본 문제의 해결 방안이 정말 아름답습니다. 바로 sigaction 함수입니다. 항상 함수를 배울 때 왜 배우냐가 중요하기 때문입니다. Sigaction 에는 sa\_mask 라는, signal 들을 block 해 주는 기능이 들어 있다.

우리가 해줄 것은 :

struct sigaction sigint;

sigint.sa\_handler = sig\_int;

sigemptyset(&sigint. sa\_mask);

sigint.sa\_flags = 0;

**sigaddset(&sigint.sa\_mask, SIGALRM);**

본 프로그램에서는 중간에 “구” 신호를 방해하는 SIGALRM 을 mask 해주면, 완벽하게 해결이 가능합니다.