[R E P O R T]

정보통신공학전공

200301582

김성태

1. 과제의 목적

리눅스가 지원하는 프로세스간 통신 방식 중 비 구조화되고 그 량이 많은 데이터의 전달에 용이한 pipe 기반의 프로세스간 통신과 어떤 이벤트를 알리는데 요긴하게 사용할 수 있는 signal에 대해 공부한다.

이 중 pipe는 부모와 자식 프로세스간에 사용되는 half-duplex(반이중) pipe와 임의의 프로세스간에 사용되는 named pipe에 대해 모두 공부한다.

공부한 내용을 점검하기 위해 기 작성된 ePDA 프로세스 관리프로그램을 이용하여 동영상을 네트워크를 통해 멀리 전송함과 동시에 화면에 나오게 하는 ePDA 동영상 중계 프로그램을 작성하고 동작을 확인해본다.

2. 공부한 내용

<개요>

시그널을 이용하면 UNIX 프로세스에 소프트웨어 인트럽트를 간단히 전송할 수 있다. 특성상으로 볼때 시그널은 프로세스간에 자료를 전송하는데 보다는 비정상적인 상황을 처리하는데 더욱 적합하다.

시그널 이름 : 시그널 이름들은 #define 명령을 이용하여 표준헤더파일인 signal.h에 정의되어 있다. UNIX에서 제공되는 시그널들은 몇개를 제외하고는 대부분 커널에 의해 사용된다.

1. SIGHUP : hangup 시그널. 이 시그널은 제어단말기의 연결이 끊어졌을 때 커널에 의하여 그 단말기에 연결된 모든 프로세스에 보내진다.

2. SIGINT : interrupt. 이 시그널은 사용자가 인터럽트키를 칠 때 커널에 의하여 단말기와 연결된 모든 프로세스에 보내진다. 이것은 수행중인 프로그램을 중지시키는 일반적인 방법이다.

3. SIGQUIT : quit. 이 시그널은 SIGINT와 마찬가지로 사용자가 단말기에서 종료(quit)키를 칠때

커널에 의하여 보내진다.

4. SIGILL : illegal Instruction. 이 시그널은 비정상적인 명령이 수행되려 할 때 운영체제로부터 보내진다.

5. SIGTRAP : trace trap. 이것은 ptrace 시스템과 함께 sdb나 adb등의 디버거에 의하여 사용되는 특별한 시그널이다.

6. SIGFPE : floating-point exception. 이것은 오버플로우나 언더플로우 같은 부동 소숫점 오류가 발생했을 때 커널에 의하여 보내진다.

7. SIGKILL : kill. 이것은 프로세스로부터 다른 프로세스를 종료시키기 위하여 보내지는 시그널이다.

8. SIGSIS : bad arguments to a system call. 이것은 프로세스가 정상적인 시스템 호출 오류 복구 방식에 의하여 처리될 수 없는 부적절한 인수를 시스템 호출에 보냈을 때 커널에 의하여 보내진다.

9. SIGPIPE : write on a pipe with no-one to read it. 파이프는 또 하나의 프로세스간 통신 방식이다.

10. SIGALRM : alarm clock. 이것은 타이머가 만료되었을 때 커널에 의하여 프로세스에 보내진다.

11. SIGTERM : software termination. 이 시그널은 보통의 프로그램에 의하여 사용될 수 있도록 제공되는 것이다.

12. SIGUSR1 & SIGUSR2 : SIGTERM과 마찬가지로 이것들은 커널에 의하여 사용되는것이 아니라 사용자가 원하는 목적을 위하여 사용될 수 있다.

- signal 시스템호출을 이용한 시그널처리

```
#include <signal.h>
        int func(), (*was)(), sig;
        .
        .
        was = signal(sig, func);
```

첫번째 인수 sig는 대상이되는 시그널을 지정한다. signal이 효과를 발휘하기 위해서는 대상이 되는 시그널이 도달하기 전에 호출이 수행되어야 한다.

SIGKILL을 제외하고는 앞에서 정의한 어떤 시그널도 sig가 될 수 있다. SIGKILL은 어떤 경우에도 프로세스를 종료시킬 수 있도록 하기 위하여 제외되었다. signal의 두번째 인수 func는 시그널에 대한 처리를 나타낸다. 이것은 세가지 값을 가질수 있다.

1. 정수값을 돌려주는 함수의 주소.
2. SIG_IGN 시그널을 무시하라는 의미의 특별한 기호 이름. 프로세스는 sig형의 시그널을 모두 무시한다.
3. SIG_DFL 시스템의 묵시적(default) 처리 상태로 복원하는 기호이름. 모든 표준 시그널 에 대하여 이것은 정상 또는 비정상 종료를 의미한다.

<시그널과 시스템 호출>

대부분의 경우 프로세스가 시스템 호출을 수행하는 도중에 시그널이 도착하면, 시그널은 시스템 호출이 종료할 때까지 아무런 영향을 받지 않는다. 그러나 몇몇 시스템 호출들은 다르게 처리되는데, 이들은 시그널에 의하여 인터럽트를 받는다. 이런 호출들은 느린 장치에 대한 read, write또는 open, wait 또는 pause 호출이다.

<시그널 재지정>

대부분의 시그널(SIGILL 과 SIGTRAP 제외)에 대한 처리 함수는 시그널이 포착된 후 즉시 재지정된다.

프로그램은 우선 시그널 SIGINT의 처리함수로 interrupt를 지정하고, 메시지를 출력한 후 10초 동안 정지한다. 만약 사용자가 인터럽트 키를 누르면, interrupt가 수행되면서 다른 메시지가 출력되고 sleep이 두번째로 수행된다. 그러나 만약 인터럽트 키가 또 다시 눌러지면 프로세스는 종료한다. UNIX 시그널은 누적되지 않으며 하나의 프로세스에 대하여 한순간에는 같은 종류의 시그널이 오직 하나만 존재한다. 반면 여러 종류의 시그널은 동시에 존재할 수 있다.

<alarm 시스템 호출>

alarm은 프로세스의 알람시계를 조작하는 간단하고도 유용한 시스템 호출이다. 타이머가 종료된 것을 프로그램에 알려주기 위하여 시그널이 사용된다. alarm은 프로세스의 수행을 중지시키는 sleep와는 다르며 대신에 즉시 복귀하여 SIGALRM 시그널이 도착할 때까지 정상적으로 수행을 계속한다. 작동중인 알람시계는 exec 호출이 수행되더라도 계속해서 작동한다.

<pause 시스템 호출>

UNIX는 alarm과 같은 부류인 pause 시스템 호출을 제공한다. pause를 호출한 프로세스는 SIGALRM 등의 시그널이 도착할 때까지 시스템 자원을 낭비하지 않도록 수행이 중지된다.

<setjmp와 longjmp>

때때로 시그널을 받았을 때 프로그램을 이전의 상태로 복구하여야 할 때가 있다. 예를 들어 사용자가 인터럽트 키를 치면 프로그램의 주메뉴로 돌아가게 해야 한다고 하면 이것은 두개의 특수한 서브루틴 setjmp와 longjmp를 이용하여 처리될 수 있다. setjmp는 프로그램의 현재 상태를 저장하고 longjmp는 저장된 상태를 다시 복구한다.

named pipe

<기본개념>

이름을 가진 파이프는 대주분 일반 파이프와 같이 작업하지만, 몇가지 주목할만한 차이점이있다.

- 이름을 가진 파이프(Named pipes)는 파일 시스템안에 특별한 장치 파일(device special file)처럼 존재한다.
- 다른 친족관계에 있는 프로세스들도 이름을 가진 파이프(named pipe)를 통해 자료를 공유할 수 있다.
- 공유하고 있는 프로세스들에 의해 모든 I/O가 수행된 후에도 이름을 가진 파이프(named pipe)는 나중의 사용을 위해 파일 시스템에 남아 있는다.

<FIFO 동작>

FIFO의 I/O 동작은 한개의 주요한 차이점을 제외하고 본질적으로 일반 파이프와 같다. "open" 시스템 호출이나 라이브러리 함수는 물리적으로 파이프의 채널을 여는데 사용되어져야 한다. 반이중 파이프에서는 파이프가 물리적인 파일 시스템이 아닌, 커널에 존재함으로 불필요하다. 예제에서 우리는 파이프를 fopen()으로 파일을 열고, fclose()로 닫는 스트림(stream)처럼 다룰 것 이다.

<FIFO의 동작차단>

일반적으로 차단은 FIFO에서 발생한다. 바꾸어 말하면, FIFO가 읽기를 위해 열려 져 있다면,

프로세스는 다른 프로세스들이 쓰기를 위해 열려고 할 때까지 차단(block)시킬 것이다. 이러한 동작은 반대로도 수행된다.

이러한 동작들을 원하지 않는다면, 디폴트 동작 차단 기능을 사용하지 않도록 open() 호출시에 O_NONBLOCK 플래그를 사용할 수 있다.

간단한 서버의 경우, 백그라운드로 밀어내고, 거기서 차단된 채로 남겨둔다. 또 다른 방법은 또 다른 가상의 콘솔로 뛰어들어 클라이언트 쪽을 실행시키고 수행 결과를 앞,뒤로 전환시키는 것이다.

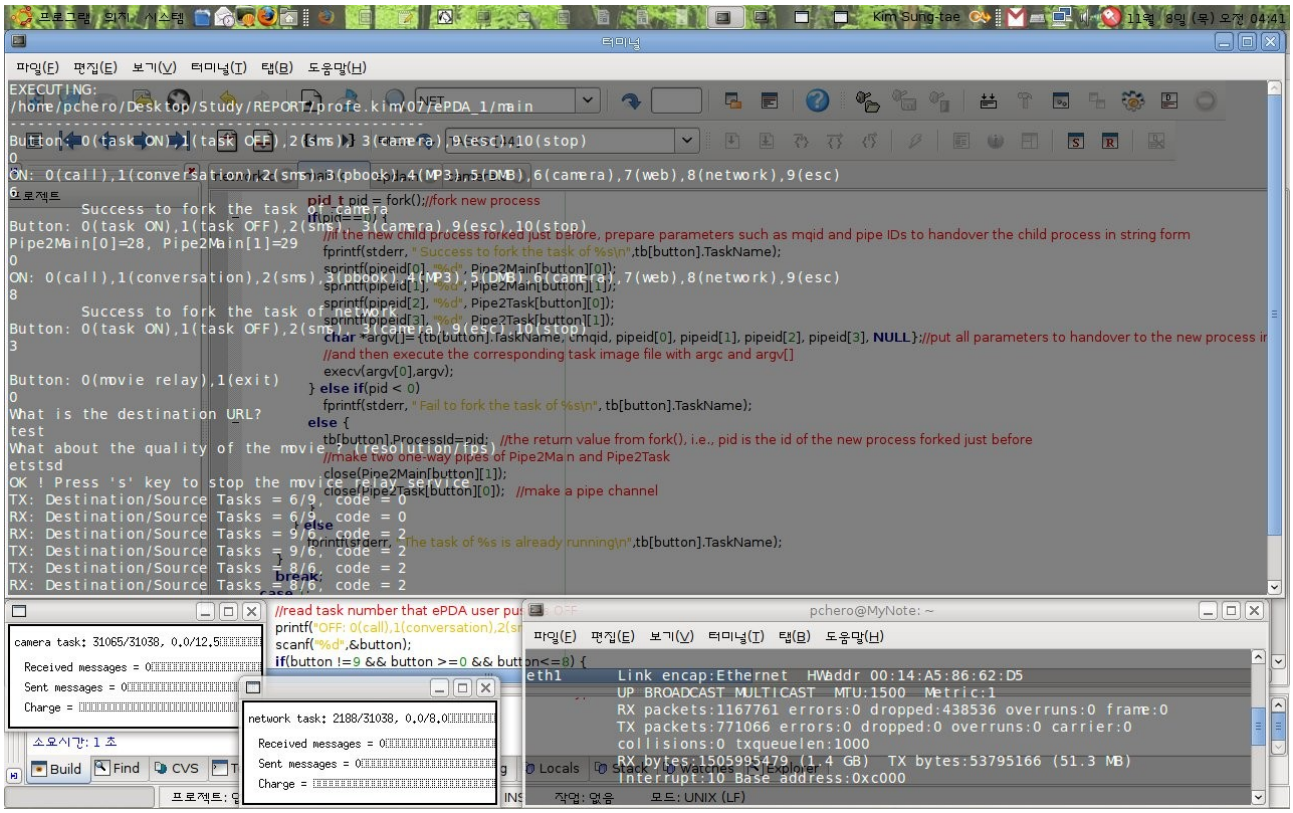
<잘 알려지지 않은 SIGPIPE 신호>

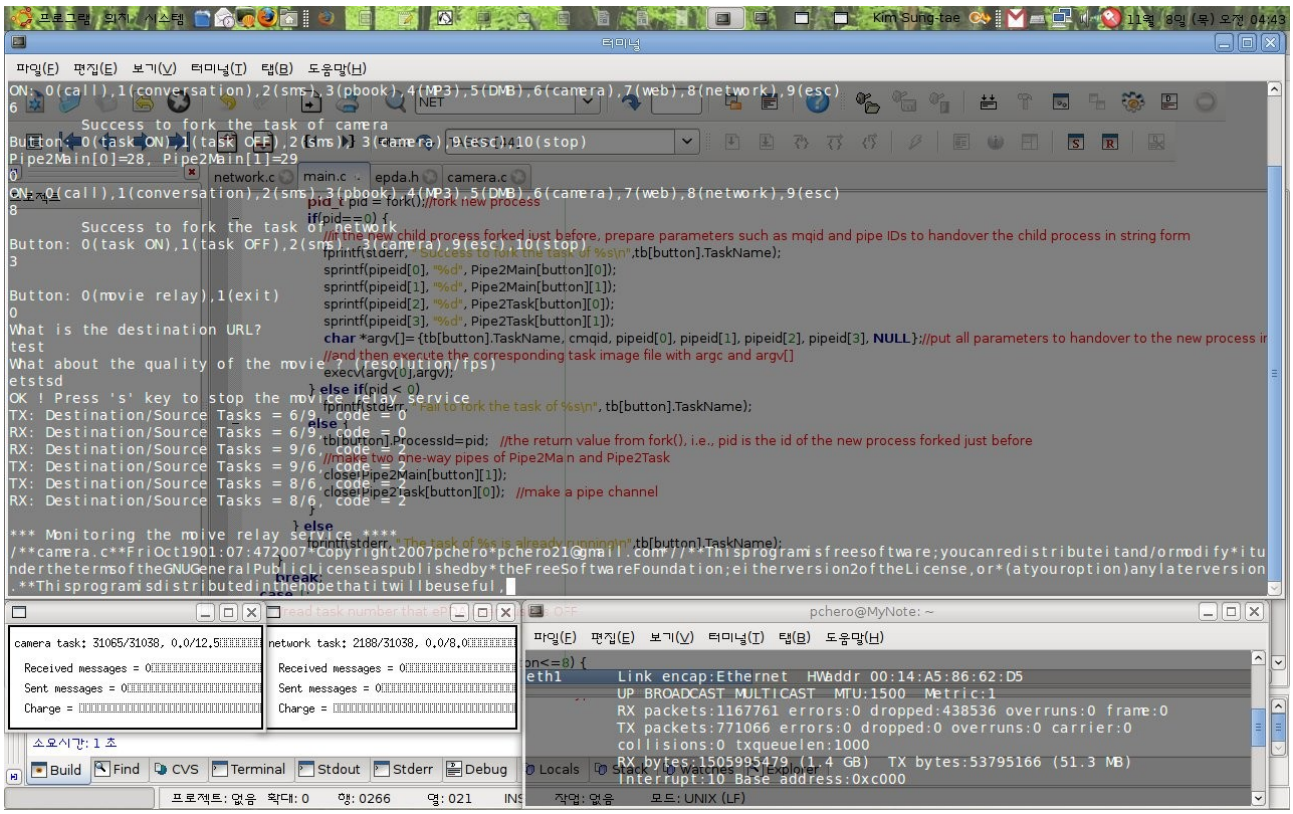
마지막 정리로, 파이프는 읽기와 쓰기를 할 수 있다.

프로세스가 읽기가 안되는 파이프에 쓰려고 한다면, 커널로 부터 SIGPIPE 신호를 받게 된다.

이것은 두개 이상의 프로세스가 파이프라인에 포함되어 있을 때 필수적이다.

3. ePDA 동영상 중계 프로그램





소스

CAMERA.c

```
smessage.type = NETWORK;

strcpy(fifo_file, "/tmp/camera2network"); // set the file name for named pipe

mkfifo(fifo_file, S_IFIFO|0666);        //make named pipe of camera2network

strcpy(stext.fifo, fifo_file);

//fprintf(stderr, "%s", stext.fifo);

StructCopy(smessage.text, &stext, BUFSIZ);
```

```
if(msgsnd(mqid, (void *)&smessage, BUFSIZ,0) == -1) {

        fprintf(stderr, "Fail to send message !!");

        exit(-1);

}

MessageLog(flog, "TX", smessage);

nsmessage++;



// emulate the movice relay service with dumping repeatly the source file of camera.c

CameraState = RELAY;

fs = fopen("camera.c", "r");

fo = fopen(fifo_file, "w");
```

```
while (CameraState == RELAY) {

        sleep(1);

        if(fscanf(fs, "%s", &buf) == EOF) {

                fclose(fs);

                fs = fopen("camera.c", "r");

        }

        //fprintf(stderr, "%s", buf);

        write(Pipe2Main, buf, BUFSIZ);

        fputs(buf, fo);

}

fclose(fs);

fclose(fo);
```

```
        //send the movie relay finish message to the main task

        smessage.type=MAIN;

        stext.SourceTaskNumber=CAMERA;

        stext.code=FINISH;

        StructCopy(smessage.text,&stext,BUFSIZ);

        //Send the prepared message

        if(msgsnd(mqid,(void *)&smessage,BUFSIZ,0)==-1) {fprintf(stderr, "Fail to send
message !!");exit(-1);}

        MessageLog(flog,"TX",smessage);nsmessage++;




        //send the movie relay finish message to the network task
```

```
        smessage.type=NETWORK;


        StructCopy(smessage.text,&stext,BUFSIZ);


        if(msgsnd(mqid,(void *)&smessage,BUFSIZ,0)==-1) {fprintf(stderr, "Fail to send
message !!");exit(-1);}


        MessageLog(flog,"TX",smessage);nsmessage++;
```

4. 수행방법

임정훈, 방영배, 김용설, 김윤겸 으로 조를 이루어 각각의 역할을 분담하여 작성하였음.

5. 요구 등급

- A 등급