Report

System Programing

Assignment #2

수 업	시스템프로그래밍실습
수업시간	월 1, 2
O S	Ubuntu 12.04 64bit
과제번호	Assignment #2
제출날짜	2014-05-23
전 공	컴퓨터공학과
학 번	2009720151
이 름	이재해

<Program Overview>

-프로그램소개

이 프로그램은 지난 Assignment #1에 Socket algorithm을 사용한 프로그램입니다. Assignment #1에서는 'tee'를 이용해 데이터를 server에 전송하고 출력하는 기능이 다였지만, 이번에는 Socket algorithm을 이용해서 데이터를 주고 받는게 가능합니다.

또한 server는 signal을 이용해서 parent processor와 child processor간의 조작이 가능합니다.

기본적으로 지난 과제에서 사용했던 명령어들의 사용이 가능합니다. ls, cd, pwd 등

추가로 종료 명령어인 quit이 추가되었습니다.

client은 종료 명령어와 control + c를 통해서 종료가 가능합니다.

이 외에도 fork() 함수를 이용해서 server의 processor를 복제해 하나의 client의 접속이 아닌 여러 개의 client가 접속 가능한 프로그램입니다. 최대 15개의 client의 접속이 가능합니다.

이러한 기능 외에도 10초마다 현재 접속한 client를 출력하는 기능을 추가하였습니다.

-문제분석

이번 과제는 이전 Assignment #1을 이용하여 제작하는 과제였기 때문에 지난 Assignment #1에 만들었던 source를 이용하여 수정, 추가하여 제작하였습니다.

수정된 부분은 지난 과제에서 데이터를 전송하는 부분이 Socket algorithm을 이용해서 수정 되는 부분이었습니다. Socket algorithm은 주어진 강의 자료를 이용했습니다.

이번 과제에서 client는 지난 과제인 Assignment #1에서 Socket algorithm으로 작동되게 수정해주면 간단히 해결되었습니다.

간단히 설명하면 socket()으로 socket을 생성하고 connect()로 연결신호를 보내며 wirte()와 read()를 이용해서 데이터를 송수신하며 종료 신호시 종료하는 것입니다. 이러한 점은 이미 이전 practice를 충실히 했다면 가능하기에 client의 socket algorithm에 대해서는 더 이상 언급하지 않겠습니다.

더 추가된 점은 종료 명령어가 입력되거나 contorl + c가 입력되었을 경우에 SIGINT가 발생하여 sh_int() 함수가 동작하여 프로그램이 종료되도록 해야합니다.

하지만 server는 fork()를 이용해서 여러 client가 접속 가능하도록 제작하고 10초마다 접속한 client를 출력하는 부분이 이번 과제의 중점적인 부분이었습니다.

먼저 10초마다 접속한 client를 출력해주는 부분도 fork()를 이용해서 따로 시간을 제기 위한 child processor를 따로 생성해서 parent processor와 signal을 이용해서 시간에 관한 정보를 주고 받도록 제작하였습니다. 이러한 방식은 낭비가 심하다고 생각되지만 다른 방식을 찾지 못하였습니다.

시간을 측정하는 방식은 struct timeval 변수와 gettimeofday() 함수를 써서 측정하였습니다. 추가로 while()문을 이용해서 계속해서 시간을 측정하였지만, 이 때에 낭비가 심하다고 생각되어 어차피 struct timeval에서 1초의 단위를 가지는 tv_sec를 이용하였기 때문에 sleep을 이용해서 1초마다 시간을 측정하도록 제작하였습니다.

정보의 전송은 signal handler를 이용해서 제작하였습니다. 또한 kill()함수를 이용해서 timer용 child processor와 parent processor를 조작하도록 만들었습니다.

다음으로 넘어가면 timer용 child processer는 parent processor가 client와 연결되는 child processor를 만들거나 시간이 10초 지나서 현재 접속된 client를 출력해줄 때 그리고 parent processor가 종료할 때를 제외하고는 서로 정보의 전송이 없습니다.

parent processor는 이제 client와 Socket algorithm을 위한 기본적인 설정을 하는데 먼저 socket()을 통해서 socket을 생성하고 bind(), listen()를 이용해서 client와의 송수신을 준비하고 client에서 신호가 들어오면 신호를 accept()하여 주는 것입니다. 이 부분에 대해서도 이미 이전 practice를 충실히 했다면 충분히 가능한 부분입니다.

이어서 말씀드리면 accept()가 되고 난 이후에 fork()를 통해서 client와 데이터를 주고받는 child processor와 현재 접속된 client의 정보들을 정리하고, client들의 정보를 출력해주며 현재 접속한 client와의 socket의 연결을 종료하는 parent로 나뉘게 됩니다.

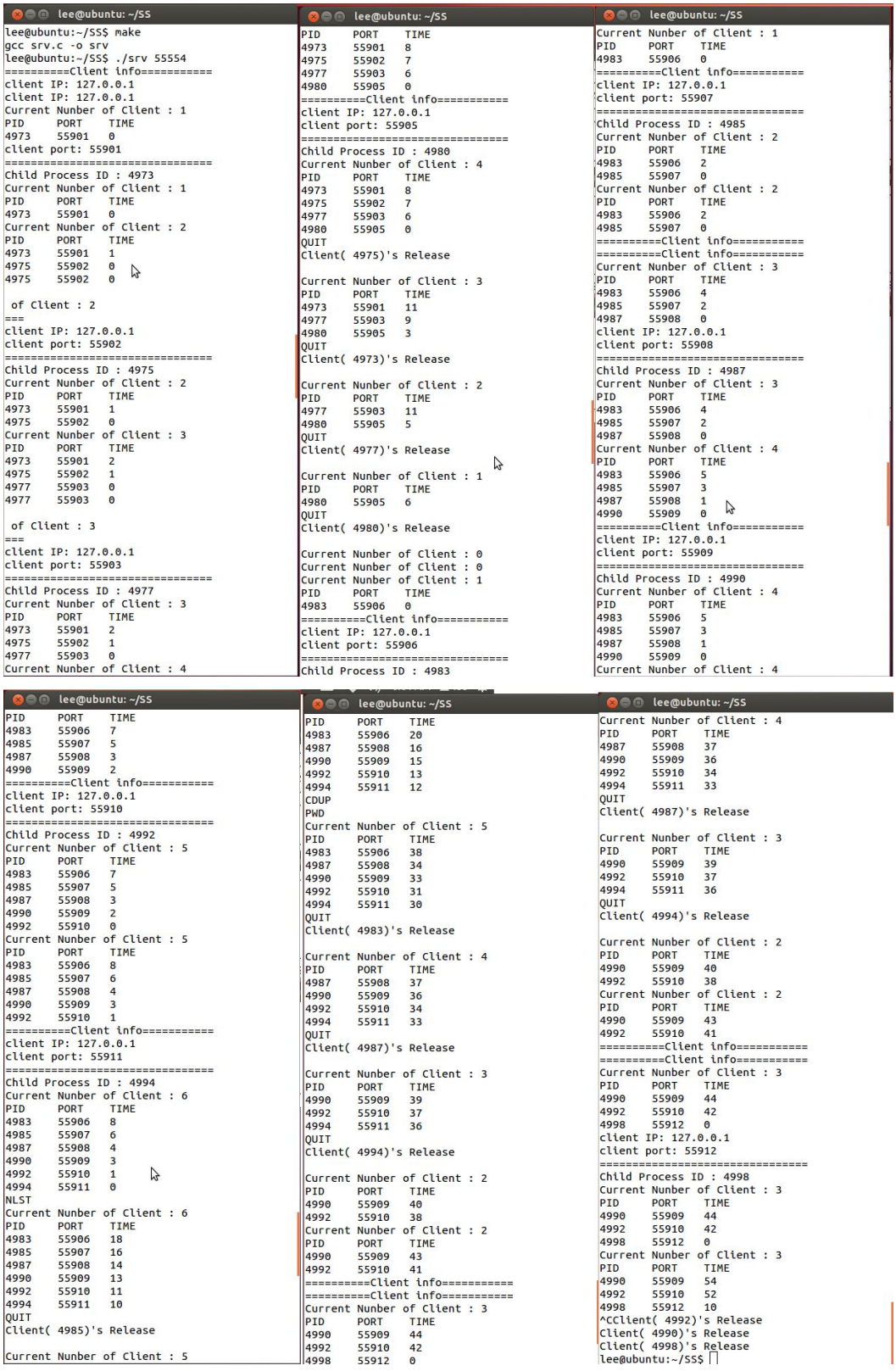
client의 정보를 저장하는 것은 새로운 linked list를 만들어서 저장하도록 제작하였습니다.

child processor에 대해서는 일반적인 socket program과 별로 다른 점이 없게 동작합니다. 단지 그 프로세서가 child processor라는 점과 client에서 종료 명령어를 입력하고 control + c를 입력하였을 경우에 현재 processor에 SIGINT가 발생하여 현재 processor가 종료된다는 것을 알리고 processor가 종료되게 됩니다.

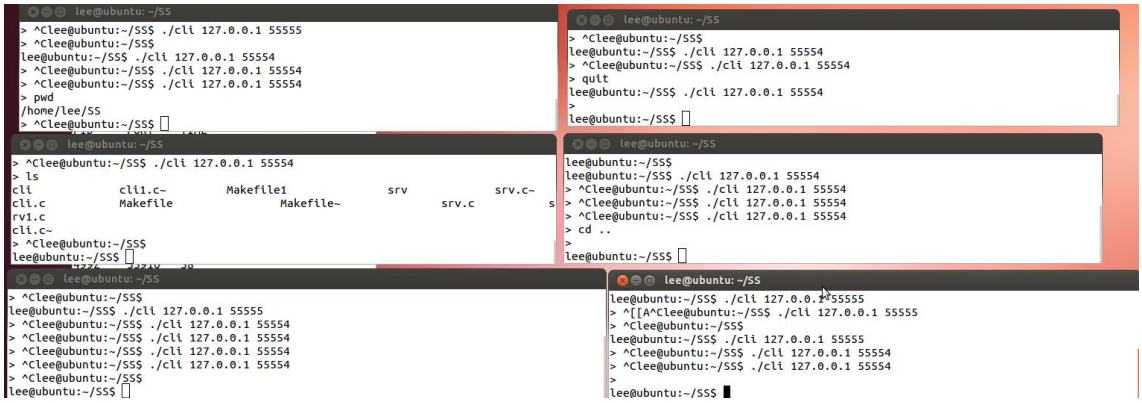
child processor가 종료되면 parent processor에서는 SIGCHLD가 발생하게 됩니다. 이를 이용해서 child processor가 종료됐다는 사실을 알고 sh_chld()함수가 동작하도록 제작하였습니다.

기본적으로 제안서에서는 SIGALRM, SIGINT, SIGCHLD를 이용하면 제작가능하다고 나왔었지만 프로그램 제작 중에 위의 signal만으로는 부족하여 임의로 SIGUSR1을 추가하여 제작하였습니다.

-실행 결과 캡쳐 화면



<server화면>



<client 화면>

<Algorithm>

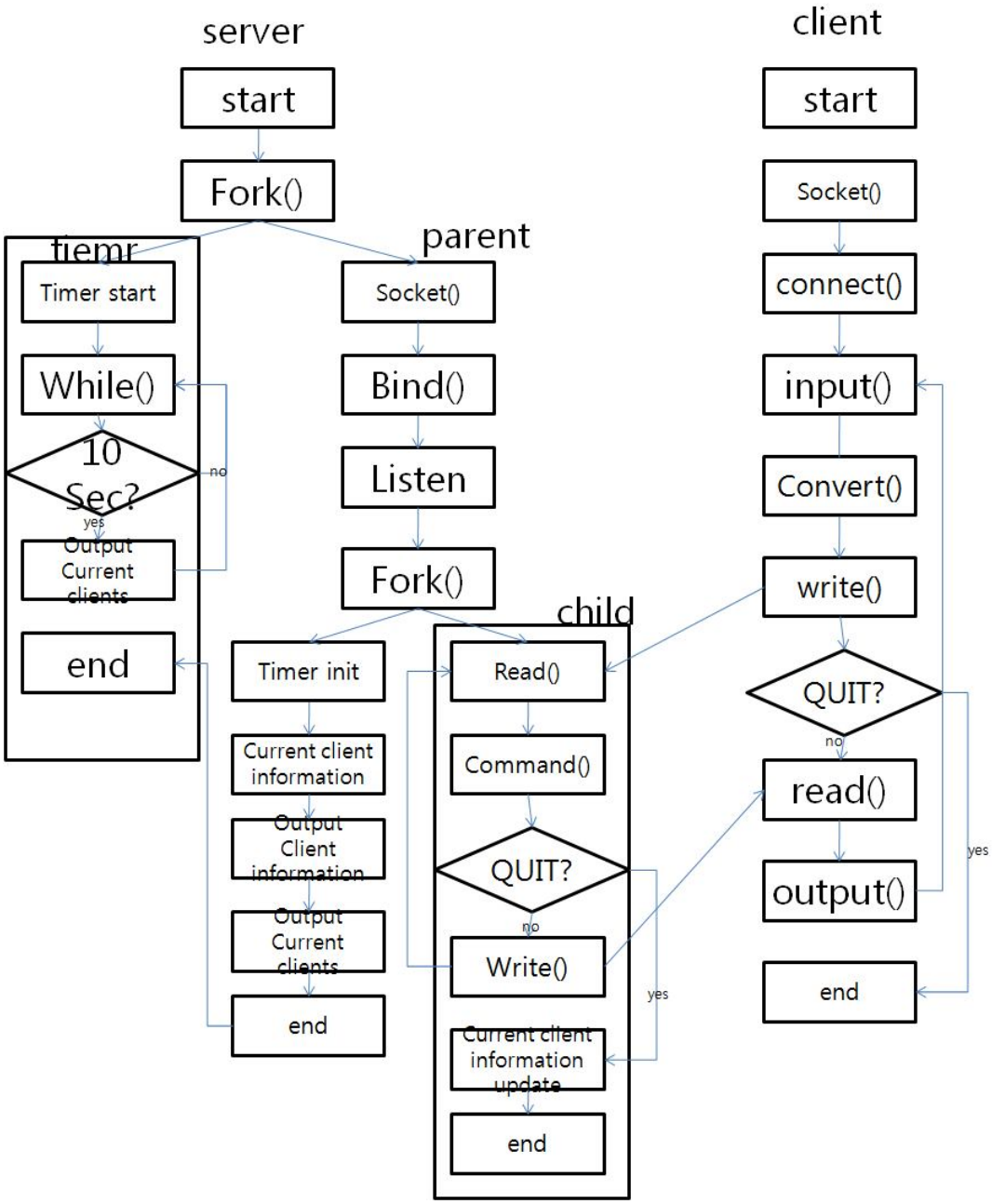
-pseudo code

```
if((Process ID = fork()) is child process)
{
        allways while
        {
                initialize buff;
                read from client_fd to buff;

                command(buff) execute;
                if(buff is 'QUIT')
                        Generation SIGINT;

                n = string(buff) lenght;
                write from buff to client_fd;
        }
}
else //parent process
{
        timer process's time update;
        now is current time;
        linked list add;
        client_cnt++;
        output client information;
        current executing clients inforamtion;
}
```

-flow chart


server
start
Fork()
tiemr
Timer start
While()
10 Sec?
no
yes
Output Current clients
end
parent
Socket()
Bind()
Listen
Fork()
Timer init
Current client information
Output Client information
Output Current clients
end
child
Read()
Command()
QUIT?
no
Write()
yes
Current client information update
end
client
start
Socket()
connect()
input()
Convert()
write()
QUIT?
no
read()
output()
end
yes

<Conslusion and discussion>

이번 과제는 지난 Assignment #1에서 Socket Algorithm을 추가하여서 제작하는 것이었습니다. 기본적으로 Assignment #2에 앞서 Practice #2-n을 진행해 오면서 Socket Algorithm에 대해서 충분히 인식하고 있다면 기본적으로 Assignment #2의 틀을 만드는 것은 간단한 문제였다고 생각합니다.

하지만 이번 과제를 진행하는 데에는 fork()함수와 signal의 처리를 제대로 인식하고 있어야지 과제의 제안서에서 요구하는 사항들을 구현할 수 있었습니다.

단지 이번 과제에서 화면에 출력하는 부분에서 문제가 발생할 수 있습니다. 이전 수업시간에 들었던 내용에 의하면 화면에 출력할 때 버퍼에 데이터가 남아있어서 그 데이터가 나중에 다시 출력된다고 들었던 부분을 기억하는데 그러한 모습이 프로그램을 실행하는 도중에 나타났습니다. 분명히 이전에 끝난 출력이 다시 한번 나오거나 일부분 끊겨서 나오는 것을 확인하였습니다.

프로그램을 천천히 동작시키면 위에서 설명한 모습이 보이지 않지만 어느 정도 빠르게 돌리게 되면 화면에서 원래는 나오지 않았던 출력들이 나오는 모습이 보입니다.

또한 프로그램을 실행시키고 10초마다 한번씩 client의 정보를 출력해주기 위해서 저는 timer용으로 child processor를 만들게 제작하였습니다. 원래는 이렇게 구성하면 안되는 것 같지만 아직 부족한 점이 많아서 위의 방법 외에는 찾지 못하였습니다. 이러한 방식은 낭비가 심하다고 생각하지만 제안서에서 요구하는 사항을 맞추기 위해서 위의 방법을 사용하였습니다.

또한 제안서에서는 SIGINT, SIGALRM, SIGCHLD를 이용하면 프로그램의 제작이 가능하다고 하였지만 저는 위의 signal로만 제작하지 못하고 SIGUSR1을 추가하여 제작하였습니다.

이번 과제는 지난 Assignment #1을 이용하였기 때문에 프로그램의 제작이 반 이상 완성된 상태로 시작했다고 볼 수 있습니다. 지난 과제에서 수행하는 명령어들을 Socket Algorithm을 이용해서 처리해주고 거기서 processor를 나누어 처리하는 과정을 익히고, signal의 처리를 알 수 있었습니다. socket algorithm에 대해서는 이미 이전에 몇 번 해보았기 때문에 문제가 없었지만 아직 signal의 처리와 fork를 이용하는 방식에 문제 많다고 생각됩니다. 다음 과제를 제작하기 전에 더 공부해서 지금 하는 방식보다 더 낳은 방식을 이용하여 제작하도록 하겠습니다.