소스코드

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <fcntl.h>
#define READ 0
#define WRITE 1

void chldsignal();
pid_t pid1,pid2;
int fg=0,list_count=0,back_ps[1024];
char back_cmd[1024][1024];

int main(int argc, char* argv[])
{
	int count=0,fd,pip_fd[2],status2,amper=0;
	char str[1024];
	char *command1,*command2,*full_cmd[1024],*full_cmd2[1024];

	signal(SIGCHLD, chldsignal);
	while(1){
		amper=0; //초기화
		printf("[prompt] "); // prompt띄우기
		fgets(str,sizeof(str),stdin);
		if(strlen(str)==1)  continue;  //  아무것도  입력하지  않고  엔터를  치면  바로
prompt 띄우기
		str[strlen(str)-1]='\0';
		//exit 또는 logout 입력하면 프로그램 종료
		if(strcmp(str,"exit")==0 || strcmp(str,"logout")==0)	exit(0);
		//background 명령이면 amper=1로 마킹
		if(str[strlen(str)-1]=='&')	amper=1;
		else fg=1; //foreground 명령이면 fg=1로 마킹
		command1=strtok(str,"&"); //마지막 & 제외한 명령어

		// myjobs 명령: 현재 돌아가고 있는 background 리스트 출력(pid,명령어)
		if(strcmp(command1,"myjobs")==0){
			for(int i=0;i<list_count;i++){
				printf("pid : %d	",back_ps[i]); // 프로세스 ID 출력
				printf("cmd : %s\n", back_cmd[i]); // 명령어 출력
			}
```

```
                continue;
        }
        pid1=fork();
        if(amper==1){ // background 명령일때
                strcpy(back_cmd[list_count],command1); // 명령어를 background 리
스트에 삽입
                back_ps[list_count++]=pid1; // pid를 background 리스트에 삽입
        }
        if(pid1==0){ //child
                if(strchr(command1,'>')!=NULL){ // command>output 일때
                        command1=strtok(command1,"> ");
                        count=0;
                        while(command1!=NULL){
                                full_cmd[count++]=command1;
                                command1=strtok(NULL,"> ");
                        }
                        command2=full_cmd[--count];
                        full_cmd[count]=(char*)0;
                        fd=open(command2,O_CREAT|O_TRUNC|O_WRONLY,0600);
//output 파일의 디스크립터
                        dup2(fd,1); //표준출력 포인터를 fd가 가리키는 곳으로 복사
                        close(fd); //fd는 닫기
                        execvp(full_cmd[0],full_cmd); //명령어 실행
                        exit(0);
                }
                else if(strchr(command1,'<')!=NULL){ // command<input 일때
                        command1=strtok(command1,"< ");
                        count=0;
                        while(command1!=NULL){
                                full_cmd[count++]=command1;
                                command1=strtok(NULL,"< ");
                        }
                        command2=full_cmd[--count];
                        full_cmd[count]=(char*)0;
                        fd=open(command2,O_RDONLY,0600);   //input 파일의 디스
크립터
                        dup2(fd,0); //표준입력 포인터를 fd가 가리키는 곳으로 복사
                        close(fd); //fd는 닫기
                        execvp(full_cmd[0],full_cmd); //명령어 실행
                        exit(0);
                }
                else if(strchr(command1,'|')!=NULL){ // command1|command2 일때
                        command1=strtok(command1,"| ");
                        count=0;
                        while(command1!=NULL){
                                full_cmd[count++]=command1;
```

```
                command1=strtok(NULL,"| ");
        }
        // ex) who|sort
        if(count==2){
                full_cmd2[0]= full_cmd[1];
                full_cmd[1]=(char*)0;
                full_cmd2[1]=(char*)0;
        }
        // ex) du -h|sort -nr
        else if(count==4){
                full_cmd2[0]= full_cmd[2];
                full_cmd2[1]= full_cmd[3];
                full_cmd[2]=(char*)0;
                full_cmd2[2]=(char*)0;
        }
        else if(count==3){
                //ex) ls -lt|head
                if(strchr(full_cmd[1],'-')!=NULL){
                        full_cmd2[0]=full_cmd[2];
                        full_cmd[2]=(char*)0;
                        full_cmd2[1]=(char*)0;
                }
                //ex) who|sort -nr
                else{
                        full_cmd2[0]=full_cmd[1];
                        full_cmd2[1]=full_cmd[2];
                        full_cmd[1]=(char*)0;
                        full_cmd2[2]=(char*)0;
                }
        }
        pipe(pip_fd); // 파이프 생성
        pid2=fork();
        if (pid2== 0) { //자식 프로세스
                close(pip_fd[READ]);
                dup2(pip_fd[WRITE],1); // 쓰기용 파이프를 표준출력
에 복제
                close(pip_fd[WRITE]);
                execvp(full_cmd[0], full_cmd); //명령어 실행
                exit(0);
        } else { // 부모 프로세스
                close(pip_fd[WRITE]);
                dup2(pip_fd[READ],0); // 읽기용 파이프를 표준입력
에 복제
                close(pip_fd[READ]);
                execvp(full_cmd2[0], full_cmd2); //명령어 실행
                exit(0);
```

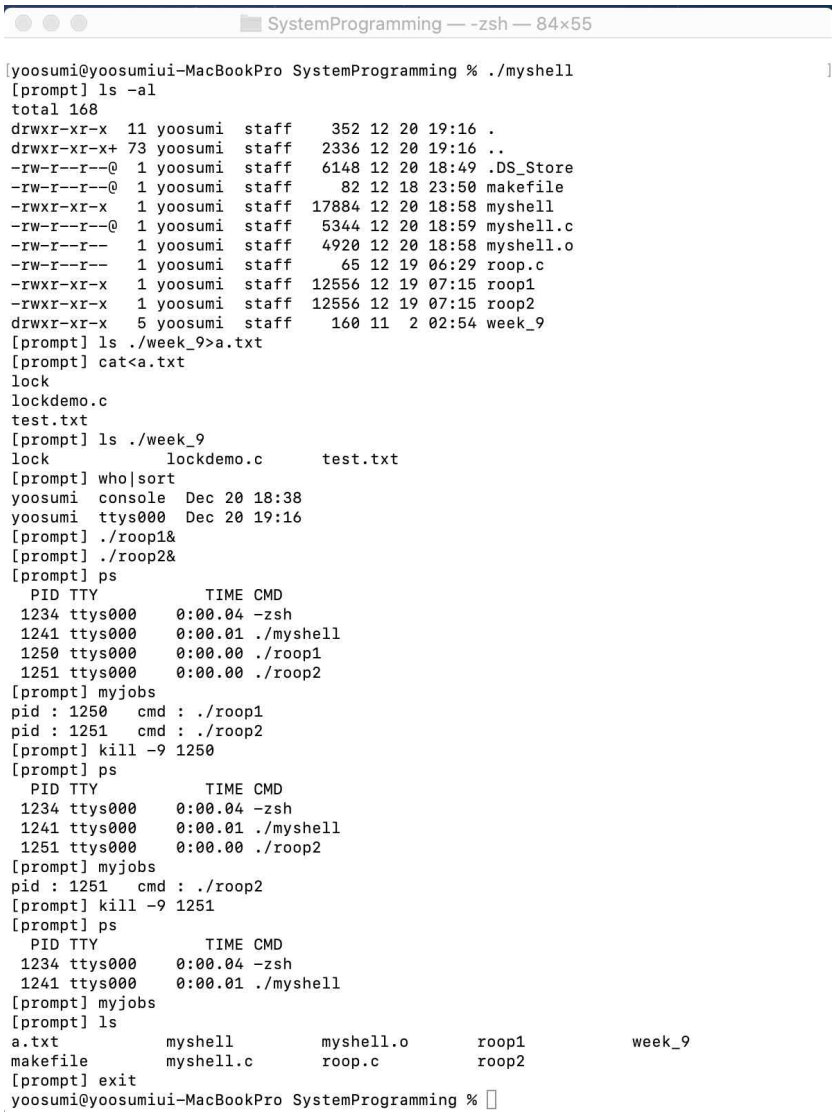
```
                        }
                    }
                    else{ // | , > , < 가 없는 명령어
                        command2=strtok(command1," ");
                        count=0;
                        while(command2!=NULL){
                            full_cmd[count++]=command2;
                            command2=strtok(NULL," ");
                        }
                        full_cmd[count]=(char*)0;
                        execvp(command1,full_cmd); //명령어 실행
                        exit(0);
                    }
                }
                else{ //parent
                    // foreground의 실행이 끝날때까지 기다린다.
                    // foreground의 실행이 끝나면(fg=0) while문을 빠져나온다.
                    while(fg==1){
                        pause();
                    }
                }
            }
}
void chldsignal()
{
        int pid, status,index1=-1,index2=-1;

        pid = waitpid(-1, &status, 0); // 임의의 자식 프로세스를 기다림

        // background 리스트에 종료된 자식 프로세스의 pid가 있는지 확인한다.
        for(int i=0;i<list_count;i++){
            if(pid==back_ps[i])
                index1=i;
        }
        // 종료된 자식 프로세스의 pid가 background 리스트에 없다면 fg=0으로 마킹한다.
        if(index1==-1){
            fg=0;
        }
        // 종료된 자식 프로세스가 background이면 background 리스트에서 삭제한다.
        else{
            if(list_count==1){
                //strcpy(back_cmd[0],NULL);
                back_ps[0]='\0';
                list_count--;
            }
            for(int j=0;j<list_count;j++){
```

```
                        if(j!=index1){
                                back_ps[++index2]=back_ps[j];
                                char copy[1024];
                                strcpy(copy,back_cmd[j]);
                                strcpy(back_cmd[index2], copy);
                        }
                }
                list_count=++index2;
        }
}
```

실행결과



```
SystemProgramming — -zsh — 84×55

yoosumi@yoosumiui-MacBookPro SystemProgramming % ./myshell
[prompt] ls -al
total 168
drwxr-xr-x  11 yoosumi  staff    352 12 20 19:16 .
drwxr-xr-x+ 73 yoosumi  staff   2336 12 20 19:16 ..
-rw-r--r--@  1 yoosumi  staff   6148 12 20 18:49 .DS_Store
-rw-r--r--@  1 yoosumi  staff     82 12 18 23:50 makefile
-rwxr-xr-x   1 yoosumi  staff  17884 12 20 18:58 myshell
-rw-r--r--@  1 yoosumi  staff   5344 12 20 18:59 myshell.c
-rw-r--r--   1 yoosumi  staff   4920 12 20 18:58 myshell.o
-rw-r--r--   1 yoosumi  staff     65 12 19 06:29 roop.c
-rwxr-xr-x   1 yoosumi  staff  12556 12 19 07:15 roop1
-rwxr-xr-x   1 yoosumi  staff  12556 12 19 07:15 roop2
drwxr-xr-x   5 yoosumi  staff    160 11  2 02:54 week_9
[prompt] ls ./week_9>a.txt
[prompt] cat<a.txt
lock
lockdemo.c
test.txt
[prompt] ls ./week_9
lock            lockdemo.c      test.txt
[prompt] who|sort
yoosumi  console  Dec 20 18:38
yoosumi  ttys000  Dec 20 19:16
[prompt] ./roop1&
[prompt] ./roop2&
[prompt] ps
  PID TTY           TIME CMD
 1234 ttys000    0:00.04 -zsh
 1241 ttys000    0:00.01 ./myshell
 1250 ttys000    0:00.00 ./roop1
 1251 ttys000    0:00.00 ./roop2
[prompt] myjobs
pid : 1250   cmd : ./roop1
pid : 1251   cmd : ./roop2
[prompt] kill -9 1250
[prompt] ps
  PID TTY           TIME CMD
 1234 ttys000    0:00.04 -zsh
 1241 ttys000    0:00.01 ./myshell
 1251 ttys000    0:00.00 ./roop2
[prompt] myjobs
pid : 1251   cmd : ./roop2
[prompt] kill -9 1251
[prompt] ps
  PID TTY           TIME CMD
 1234 ttys000    0:00.04 -zsh
 1241 ttys000    0:00.01 ./myshell
[prompt] myjobs
[prompt] ls
a.txt           myshell         myshell.o       roop1           week_9
makefile        myshell.c       roop.c          roop2
[prompt] exit
yoosumi@yoosumiui-MacBookPro SystemProgramming %
```