Operating Systems(24802)
project #1

소프트웨어학부
2019037129
신민경

목차.

1. 구현 목표

* 올바르게 들어온 입력에 대해 명령어를 수행할 수 있어야 한다.
* &이 명령어 뒤에 오면 백그라운드에서 실행할 수 있어야 한다.
* |이 있으면 왼쪽 명령어의 출력이 오른쪽 명령어의 입력으로 사용되어야 한다.
* <, >이 있으면 파일을 redirection 할 수 있어야 한다.
* (추가목표) |, &을 이용하여 명령어 여러 개가 한 번에 들어왔을 때 수행할 수 있어야 한다.
* (추가목표) |와 &을 같이 처리할 수 있어야 한다.

2. Simple Shell Algorithm

1. 한 줄로 된 입력을 받는다. (get_args 함수)
2. “ ”, “\n”을 기준으로 문자열을 쪼개 args에 저장한다. (parse_args 함수)
3. exit 커맨드가 있으면 프로그램을 종료한다.
4. fork()를 이용하여 자식 프로세서를 만들고 자식프로세서에서 왼쪽에서부터 커맨드를 찾아 수행한다.
5. 커맨드를 찾아 수행하는 규칙은 다음과 같다.
    5-1. args[0]은 커맨드이다.
    5-2. ‘&’, ‘|’의 오른쪽 값은 커맨드이다.

5-3. ‘&’, ‘|’을 만날 때까지 커맨드 위치에서부터 오른쪽으로 args를 읽는다.
5-4. ‘<’, ‘>’, ‘&’, ‘|’을 만나면 args[i]를 NULL로 바꿔준다.
5-5. ‘<’. ‘>’을 만나면 redirection 해준다.
5-6. ‘&’, ‘|’을 만나면 새로운 자식 프로세서를 fork하고 자식에서 바로 이전 커맨드를 수행한다.
5-5-1. ‘&’이면 부모 프로세서에서는 wait을 하지 않고 바로 다음 커맨드를 찾는다.
5-5-2. ‘|’이면 부모 프로세서에서는 자식 프로세서의 결과 값을 dup2를 이용하여 input으로 받고 다음 커맨드를 찾는다.
5-7. 더 이상 커맨드가 없을 때까지 5-1부터의 과정을 반복한다.
5-8. 더 이상 커맨드가 없으면 부모 프로세서가 종료된다.

3. 프로그램 소스코드

```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <sys/wait.h>
#define MAX_LINE 80 /* The maximum length command */

// 연산자 ID 부여
int operator_id(char *str){
    if(strcmp(str,">") == 0) return 1;
    if(strcmp(str, "<") == 0) return 2;
    if(strcmp(str, "|") == 0) return 3;
    if(strcmp(str, "&") == 0) return 4;
    return 0;
}

 // '<', '>'을 만나면 file redirecting
void redirecting(char* file, int id){
    if(id == 1){ // ‘>’
        int fd = open(file, O_WRONLY|O_CREAT, 0666);
        dup2(fd,STDOUT_FILENO);
        close(fd);
    }
    else if(id == 2){ // ‘<’
        int fd = open(file, O_RDONLY, 0666);
        dup2(fd,STDIN_FILENO);
        close(fd);
    }
}
```

```c
/* command_pos의 바로 다음 명령어를 찾는 함수
    더 이상 실행할 명령어가 args에 없으면 종료.

    command_pos부터 오른쪽으로 탐색하며 '<', '>', '&', '|'을 만나면 NULL로 바꿔주고
    각각 '<', '>'은 redirection, '&', '|'은 새로 fork하여 자식 프로세서에서 command_pos 위치의 명령어를 실행한다.
    '&', '|' 바로 오른쪽의 args 값은 명령어이기 때문에 다음 명령어 자리로 one_command를 다시 호출한다.
 */
void one_command(char **args, int command_pos, int size){
    if(command_pos == size) return;
    int background = 0, is_pipe = 0, next_command = size;

    for(int i=command_pos; i<next_command; i++){
        int id = operator_id(args[i]);

        if(!id) continue;
        args[i] = NULL;

        if(id <= 2) redirecting(args[i+1],id); // <, >
        else{
            if(id == 3) is_pipe = 1; // pipe
            else if(id == 4) background = 1; // &
            next_command = i+1; // the right of '&', '|' is command
        }
    }

    if(is_pipe){
        int state, fd[2];
        pipe(fd);
        pid_t pid = fork();

        // pipe(A|B)
        if(pid < 0){
            fprintf(stderr, "Fork failed");
            return;
        }
        else if(pid > 0){ // parent(B)
            waitpid(pid, &state, 0);

            dup2(fd[0],STDIN_FILENO);
            close(fd[1]);
            one_command(args, next_command, size);
        }
        else{ // child(A)
            dup2(fd[1],STDOUT_FILENO);
            close(fd[0]);

            execvp(args[command_pos], &args[command_pos]);
        }
        return;
    }
```

```
    else{
        int state;
        pid_t pid;
        pid = fork();

        if(pid < 0){
            fprintf(stderr, "Fork failed");
            return;
        }
        else if(pid > 0){ // parent
        // if background is false, wait child process terminated.
            if(!background) waitpid(pid, &state, 0);
            one_command(args, next_command, size);
        }
        else if(pid == 0){ // child
            execvp(args[command_pos], &args[command_pos]);
        }
        return;
    }
}

// delimiter를 기준으로 args 문자열을 자름
int parse_args(char* istream, char** args, char *delimiter){
    char *token = strtok(istream, delimiter);

    int cnt = 0;
    while(token!=NULL){
        args[cnt++] = strdup(token);
        token = strtok(NULL, delimiter);
    }

    args[cnt] = NULL;
    return cnt;
}

// args를 읽고 " ", "\n"을 기준으로 파싱한다.
int get_args(char** args){
    char istream[MAX_LINE];

    fgets(istream, sizeof(istream), stdin);
    fflush(stdout);

    return parse_args(istream, args, " \n");
}

int main(void){
    char *args[MAX_LINE/2 + 1]; /* command line arguments */
    int should_run = 1; /* flag to determine when to exit program */

    while(should_run) {
```

```
        printf("osh>");
        int size = get_args(args);
        if(size && strcmp(args[0], "exit") == 0) break;

        int state;
        pid_t pid;
        pid = fork();

        if(pid < 0){
            fprintf(stderr, "Fork failed");
            break;
        }
        else if(pid > 0){ // parent process
            waitpid(pid, &state, 0);
            printf("Done\n");
        }
        else if(pid == 0){ // child process
            one_command(args, 0, size);
            break;
        }
    }
    return 0;
}
```

3. 컴파일 과정

```
hihiroo@hihiroo-VirtualBox:~/Desktop$ gcc -v hello.c
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/7/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 7.5.0-3ubuntu1~18.04' --with-bugurl=file:///usr/share/doc/gcc-7/README.Bugs --enable-l
anguages=c,ada,c++,go,brig,d,fortran,objc,obj-c++ --prefix=/usr --with-gcc-major-version-only --program-suffix=-7 --program-prefix=x86_64-linux-gnu-
--enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --ena
ble-bootstrap --enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object -
-disable-vtable-verify --enable-libmpx --enable-plugin --enable-default-pie --with-system-zlib --with-target-system-zlib --enable-objc-gc=auto --enab
le-multiarch --disable-werror --with-arch-32=i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with-tune=generic --enable-off
load-targets=nvptx-none --without-cuda-driver --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)
COLLECT_GCC_OPTIONS='-v' '-mtune=generic' '-march=x86-64'
 /usr/lib/gcc/x86_64-linux-gnu/7/cc1 -quiet -v -imultiarch x86_64-linux-gnu hello.c -quiet -dumpbase hello.c -mtune=generic -march=x86-64 -auxbase he
llo -version -fstack-protector-strong -Wformat -Wformat-security -o /tmp/ccSRBSDI.s
GNU C11 (Ubuntu 7.5.0-3ubuntu1~18.04) version 7.5.0 (x86_64-linux-gnu)
        compiled by GNU C version 7.5.0, GMP version 6.1.2, MPFR version 4.0.1, MPC version 1.1.0, isl version isl-0.19-GMP

GGC heuristics: --param ggc-min-expand=100 --param ggc-min-heapsize=131072
ignoring nonexistent directory "/usr/local/include/x86_64-linux-gnu"
ignoring nonexistent directory "/usr/lib/gcc/x86_64-linux-gnu/7/../../../../x86_64-linux-gnu/include"
#include "..." search starts here:
#include <...> search starts here:
 /usr/lib/gcc/x86_64-linux-gnu/7/include
 /usr/local/include
 /usr/lib/gcc/x86_64-linux-gnu/7/include-fixed
 /usr/include/x86_64-linux-gnu
 /usr/include
End of search list.
GNU C11 (Ubuntu 7.5.0-3ubuntu1~18.04) version 7.5.0 (x86_64-linux-gnu)
        compiled by GNU C version 7.5.0, GMP version 6.1.2, MPFR version 4.0.1, MPC version 1.1.0, isl version isl-0.19-GMP

GGC heuristics: --param ggc-min-expand=100 --param ggc-min-heapsize=131072
Compiler executable checksum: b62ed4a2880cd4159476ea8293b72fa8
COLLECT_GCC_OPTIONS='-v' '-mtune=generic' '-march=x86-64'
 as -v --64 -o /tmp/ccdI6oQt.o /tmp/ccSRBSDI.s
GNU assembler version 2.30 (x86_64-linux-gnu) using BFD version (GNU Binutils for Ubuntu) 2.30
COMPILER_PATH=/usr/lib/gcc/x86_64-linux-gnu/7/:/usr/lib/gcc/x86_64-linux-gnu/7/:/usr/lib/gcc/x86_64-linux-gnu/:/usr/lib/gcc/x86_64-linux-gnu/7/:/usr/
lib/gcc/x86_64-linux-gnu/
LIBRARY_PATH=/usr/lib/gcc/x86_64-linux-gnu/7/:/usr/lib/gcc/x86_64-linux-gnu/7/../../../x86_64-linux-gnu/:/usr/lib/gcc/x86_64-linux-gnu/7/../../../../
lib/:/lib/x86_64-linux-gnu/:/lib/../lib/:/usr/lib/x86_64-linux-gnu/:/usr/lib/../lib/:/usr/lib/gcc/x86_64-linux-gnu/7/../../../:/lib/:/usr/lib/
COLLECT_GCC_OPTIONS='-v' '-mtune=generic' '-march=x86-64'
 /usr/lib/gcc/x86_64-linux-gnu/7/collect2 -plugin /usr/lib/gcc/x86_64-linux-gnu/7/liblto_plugin.so -plugin-opt=/usr/lib/gcc/x86_64-linux-gnu/7/lto-wr
apper -plugin-opt=-fresolution=/tmp/ccDqAoBe.res -plugin-opt=-pass-through=-lgcc -plugin-opt=-pass-through=-lgcc_s -plugin-opt=-pass-through=-lc -plu
gin-opt=-pass-through=-lgcc -plugin-opt=-pass-through=-lgcc_s --build-id --eh-frame-hdr -m elf_x86_64 --hash-style=gnu --as-needed -dynamic-linker /l
ib64/ld-linux-x86-64.so.2 -pie -z now -z relro /usr/lib/gcc/x86_64-linux-gnu/7/../../../x86_64-linux-gnu/Scrt1.o /usr/lib/gcc/x86_64-linux-gnu/7/../.
./../x86_64-linux-gnu/crti.o /usr/lib/gcc/x86_64-linux-gnu/7/crtbeginS.o -L/usr/lib/gcc/x86_64-linux-gnu/7 -L/usr/lib/gcc/x86_64-linux-gnu/7/../../..
/x86_64-linux-gnu -L/usr/lib/gcc/x86_64-linux-gnu/7/../../../../lib -L/lib/x86_64-linux-gnu -L/lib/../lib -L/usr/lib/x86_64-linux-gnu -L/usr/lib/../l
ib -L/usr/lib/gcc/x86_64-linux-gnu/7/../../.. /tmp/ccdI6oQt.o -lgcc --push-state --as-needed -lgcc_s --pop-state -lc -lgcc --push-state --as-needed -
lgcc_s --pop-state /usr/lib/gcc/x86_64-linux-gnu/7/crtendS.o /usr/lib/gcc/x86_64-linux-gnu/7/../../../x86_64-linux-gnu/crtn.o
COLLECT_GCC_OPTIONS='-v' '-mtune=generic' '-march=x86-64'
hihiroo@hihiroo-VirtualBox:~/Desktop$
```

4. 명령어 실행 결과물 및 설명

1. ls -l

```
hihiroo@hihiroo-VirtualBox:~/Desktop$ gcc hello.c
hihiroo@hihiroo-VirtualBox:~/Desktop$ ./a.out
osh>ls -l
total 24
-rwxr-xr-x 1 hihiroo hihiroo 13360  3월 29 23:06 a.out
-rw-r--r-- 1 hihiroo hihiroo  3338  3월 28 12:04 hello.c
-rw-r--r-- 1 hihiroo hihiroo   181  3월 28 21:58 out.txt
Done
osh>
```

2. ls -l &

```
osh>ls -l &
Done
osh>total 24
-rwxr-xr-x 1 hihiroo hihiroo 13360  3월 29 23:06 a.out
-rw-r--r-- 1 hihiroo hihiroo  3338  3월 28 12:04 hello.c
-rw-r--r-- 1 hihiroo hihiroo   181  3월 28 21:58 out.txt
```

3. ls -l > out.txt

```
ls -l > out.txt
Done
osh>cat out.txt
total 24
-rwxr-xr-x 1 hihiroo hihiroo 13360  3월 29 23:06 a.out
-rw-r--r-- 1 hihiroo hihiroo  3338  3월 28 12:04 hello.c
-rw-r--r-- 1 hihiroo hihiroo   181  3월 28 21:58 out.txt
Done
osh>
```

4. ls -l > out2.txt &

```
osh>ls -l > out2.txt &
Done
osh>cat out2.txt
total 28
-rwxr-xr-x 1 hihiroo hihiroo 13360  3월 29 23:06 a.out
-rw-r--r-- 1 hihiroo hihiroo  3338  3월 28 12:04 hello.c
-rw-r--r-- 1 hihiroo hihiroo   240  3월 29 23:11 out2.txt
-rw-r--r-- 1 hihiroo hihiroo   181  3월 29 23:10 out.txt
Done
osh>
```

5. cat -n < out.txt

```
osh>cat -n < out.txt
     1  total 24
     2  -rwxr-xr-x 1 hihiroo hihiroo 13360  3월 29 23:06 a.out
     3  -rw-r--r-- 1 hihiroo hihiroo  3338  3월 28 12:04 hello.c
     4  -rw-r--r-- 1 hihiroo hihiroo   181  3월 28 21:58 out.txt
Done
osh>
```

6. cat -n < out.txt &

```
osh>cat -n < out.txt &
Done
osh>     1      total 24
     2  -rwxr-xr-x 1 hihiroo hihiroo 13360  3월 29 23:06 a.out
     3  -rw-r--r-- 1 hihiroo hihiroo  3338  3월 28 12:04 hello.c
     4  -rw-r--r-- 1 hihiroo hihiroo   181  3월 28 21:58 out.txt
```

7. sort -u < out3.txt > out4.txt

```
cat hello.c > out3.txt
Done
osh>sort -u < out3.txt > out4.txt
Done
osh>cat out4.txt



        }
        }
    }
}
    args[cnt] = NULL;
        args[cnt++] = strdup(token);
        args[i] = NULL;
            break;
    char *args[MAX_LINE/2 + 1]; /* command line arguments */
    char istream[MAX_LINE];
    char *token = strtok(istream, delimiter);
        close(fd);
            close(fd[0]);
            close(fd[1]);
#define MAX_LINE 80 /* The maximum length command */
            dup2(fd[0],STDIN_FILENO);
            dup2(fd[1],STDOUT_FILENO);
        dup2(fd,STDIN_FILENO);
```

8. ls -a | less -a

```
osh>ls -a | less -a
Done
  .
  ..
  a.out
  hello.c
  out2.txt
  out3.txt
  out4.txt
  out.txt
  .vscode
  (END)
```

9. date & ls -a

```
osh>date & ls -a
.  ..  a.out  hello.c  out2.txt  out3.txt  out4.txt  out.txt  .vscode
2021. 03. 29. (월) 23:19:15 KST
Done
osh>
```

10. ls | cat -n &

```
osh>ls | cat -n &
Done
osh>     1      a.out
     2  hello.c
     3  out2.txt
     4  out3.txt
     5  out4.txt
     6  out.txt
```

11. mkdir test & ls -a

```
mkdir test & ls -a
.  ..  a.out  hello.c  out2.txt  out3.txt  out4.txt  out.txt  .vscode
Done
osh>ls
a.out  hello.c  out2.txt  out3.txt  out4.txt  out.txt  test
Done
osh>
```

12. ls | cat -n | less

```
osh>ls | cat -n | less
Done
osh>
```

```
     1  a.out
     2  hello.c
     3  out2.txt
     4  out3.txt
     5  out4.txt
     6  out.txt
     7  test
(END)
```