Operating System

HW1
-Simple Shell

SeungHyun Yeo
Dankook University

2018 autumn

1. Project Introduction

간단한 shell을 만들어 shell의 역할과 동작 이해하기

2. Motivation

Shell은 운영체제의 커널과 사용자 사이를 이어주는 역할을 하는 프로그램으로서 커널과 사용자의 가운데 중간다리 역할을 한다. 그렇기 때문에 Shell은 사용자의 명령어를 입력받으면 해당 명령어를 해석하고 ,명령어 수행하는 형식을 갖춘다. Shell은 이러한 방식으로 Operating System에서 사용자의 명령어 실행이 가능하게 하기 위해 주로 low-level OS functions들을 사용한다.

3. Concepts used in Shell simulator

fork()

fork는 child process를 만드는데 사용된다. fork를 함수를 call 하면 parent process에서는 child process의 process id가 반환되고 child process에서는 0이 반환된다. 실패할 경우에는 -1이 반환된다.

stat()

stat함수로는 파일의 상태를 확인할 수 있다. 첫 번째 parameter로 파일이름을 받고 두 번째 parameter로 파일의 정보 등을 저장하는 함수이다. 또한 함수의 반환 값으로 파일 정보를 성공적으로 가져왔을 경우 0이 반환되고 그렇지 않은 경우 -1이 반환된다.

execve()

실행 가능한 파일을 현재 프로세서에서 실행하여 기존에 실행되던 프로그램을 교체한다. execve() 함수의 경우 3개의 parameter를 입력받는데 첫 번째로는 파일이름을 입력받는다. 이때 파일이름은 정확한 위치를 지정해주어야 한다. 두 번째로는 string array로 첫 번째 스트링에는 첫 번째 parameter와 동일한 값, 그 다음부터는 첫 번째 parameter로 받은 프로그램이 실행될 때 인자로 받을 수 있는 parameter를 받는다. 마지막은 항상 NULL로 끝나야 한다. 세 번째 parameter로는 환경변수 string array로 마지막은 NULL이어야 한다.

4. Program Structure

get PATH
loop until get“quit”
    1. read string from keyboard input
    2. loop until stat() return 0
        1. make absolute path
        2. check stat()’s return value
    3. end loop
    4. fork()
    5. if child process
        1. execve()
        2. terminate child process
    6. else parent process
        1. wait for completeness of child process
end loop

5. Problem and Solution

첫 번째로 직면했던 문제는 getenv(), fork(), execve(), stat()함수에 대해서 잘 모른다는 것이었다. 그나마 getenv()나 fork(), stat()의 경우에는 예제코드가 주어져서 각 예제코드들을 직접 실행 해보면서 어떤 값을 반환하고 어떤 parameter를 받는지 확인하였다. 하지만 execve의 경우에는 예제 코드가 주어지지 않아서 manual page에 있는 코드를 복사해서 직접 실행함으로써 execve함수에 관해서 조금 알게 되었다.

두 번째로 직면 했던 문제는 키보드 인풋을 getenv를 통해서 받아온 경로와 strcat를 통하여 프로그램의 absolute path를 만들 때 발생하였다. 기존 path에 /를 붙이고 input을 붙이는 방식을 택했는데 짝수 번째 string이 사라지는가 하면 /나 input이 두 번식 붙여지는 경우도 발생을 했는데 원인은 잘 모르겠지만 char *에 동적 할당을 했더니 문제가 사라짐일 미루어 볼 때 아마도 메모리 할당관련 문제였던 것 같다.

세 번째로는 child process를 없애지 않아서 forkbomb이 일어났다. 이로 인해서 서버에서 계속해서 프로세스가 생겨나는 일 이 발생했으며 child process에 exit()을 추가해서 해결했고 forkbomb은 프로세스를 지워도 계속해서 생겨나서 결국 서버는 reboot했다.

네 번째로는 직면했던 문제는 keyboard입력을 받는 경우 발생했다. 초기버전에서는 gets() 함수를 사용했었는데 warning message가 계속 떠서 확인결과 gets()함수는 boundary check를 하지 않아 다른 메모리 영역으로의 overflow 발생 가능성이 있어서 man page에서도 절대 사용하지 말라고 써져있어서 fgets()함수로 대체하게 되었다.

6. Additional

몇 가지 추가구현 사항을 구현하였습니다. 우선 make 명령어를 통해서 executable file을 생성 할 수 있도록 shell script를 작성하였습니다. 뿐만 아니라 getenv("PWD")를 통해서 PWD에 대한 정보를 출력할 수 있게 하였습니다. TIME을 입력으로 받게 되면 현재 시간을 시간 분 초 단위로 출력하게 되는데 이의 경우 time()이라는 함수를 이용하여 현재시간에 관한 정보를 구조 체에 받아와서 출력하는 방식으로 구현을 했습니다. USER를 입력 하면 getenv("USER")을 통해서 user의 이름을 출력하도록 하였습니다. 추가적으로 change directory 또한 chdir() 함수를 통해서 구현하였습니다. ls의 경우 argument를 프로그램에 추가적으로 넘겨줄 수 있도록 하여 예를 들어 ls -a, ls -alh 등이 실행 될 수 있도록 하였습니다.

7. Build environment

Compilation : Linux, with GCC

8. Code

```
28 int main()
29 {
30          char command[20];
31          char *input = (char*)malloc(sizeof(char)*50);
32          char *argv[]={NULL,NULL,NULL,NULL,NULL};
33
34          cgetenv();
35          while(1)
36          {
37                  getcwd(buf,255);
38                  env=getenv("USER");
39                  printf("SISH %s@assam:%s : ",env,buf);
40                  fgets(command,sizeof(command),stdin);
41                  command[strlen(command)-1]='\0';
42                  if(strncmp("quit",command,4)==0)
43                  {
44                          exit(0);
45                  }
46                  else if(strncmp("PWD",command,3)==0)
47                  {
48                          env = getenv("PWD");
49                          printf("%s\n",env);
50                  }
51                  else if(strncmp("USER",command,4)==0)
52                  {
53                          env =getenv("USER");
54                          printf("User name :%s\n",env);
55                  }
56                  else if(strncmp(" ",command,1)==0)
57                  {
58                          continue;
59                  }
60                  else if(strncmp("TIME",command,4)==0)
61                  {
62                          time_t timer;
63                          struct tm *t;
64                          timer = time(NULL);
65                          t = localtime(&timer);
66                          printf("Current Time : %d:%d:%d(H/M/S)\n",t->tm_hour,t->tm_min,t->tm_sec);
67                  }
```

```
68              else if(strncmp("cd ",command,3)==0)
69              {
70                      strtok_r(command," ",&saveptr);
71                      chdir(saveptr);
72              }
73              else
74              {
75                      strtok_r(command," ",&saveptr);
76                      if((strncmp("/",command,1))!=0)
77                      {
78                              for(j=0;tok[j]!=NULL;j++)
79                              {
80                                      sprintf(input,"%s/%s",tok[j],command);
81                                      ret=stat(input,&fstat_buf);
82                                      if(ret==0)break;
83                              }
84                      }
85                      else
86                      {
87                              sprintf(input, "%s",command);
88                              ret=stat(input, &fstat_buf);
89                      }
90                      if(ret==-1)
91                      {
92                              printf("No such file : %s\n",command);
93                              continue;
94                      }
95
96                      pid=fork();
97                      if(pid==-1)
98                      {
99                              perror("fork error");
100                     }
```

```
101                         else if(pid==0)
102                         {
103                                 printf("****child pid : %d****\n",getpid());
104                                 char *newenviron[]={NULL};
105                                 argv[0]=input;
106                                 for(j=1;;j++)
107                                 {
108                                         argv[j]=strtok_r(saveptr," ",&saveptr);
109                                         if(argv[j]==NULL) break;
110                                 }
111                                 ret=execve(input,argv,newenviron);
112                                 if(ret==-1) printf("execve error\n");
113                                 exit(0);
114                         }
115                         else
116                         {
117                                 wait(0);
118                         }
119                 }
120         }
121         return 0;
122 }
123
124
125 int cgetenv()
126 {
127         env = getenv("PATH");
128         for (j=0,str=env; ;str= NULL,j++)
129         {
130                 tok[j] = strtok_r(str, ":", &saveptr);
131                 if (tok[j] == NULL) break;
132         }
133         return 0;
134 }
```

9. Personal feelings

Shell을 만들면서 느꼈던 점은 지난 학기 mobile processor 수업을 통해서 computer의 cpu가 어떻게 동작하는지 알게 되었다면 Shell을 만들면서 사용자의 input에 따라서 computer가 어떻게 동작하게 되는지 알게 되어 재미있었습니다. 그리고 비록 잘 알고 쓰진 못했지만 low-level OS function들도 새롭게 써보게 되어서 좋은 경험이라고 생각했습니다. 뿐만 아니라 의도치 않게 forkbomb을 발생시키게 되었는데 이와 관련해서 외부 공격뿐만 아니라 정상적으로 서버에 접근이 가능한 사람 또한 전문지식이 부족하거나 실수에 의해서 보안 위협요소가 될 수 있다고 생각했습니다. 결과적으로는 과제 자체를 이해한다면 과제의 난이도는 그렇게 높지는 않아서 재미있게 할 수 있었고 뿐만 아니라 이번 과제 이후로 이어질 과제나 수업에 대한 기대나 맥락을 잡을 수 있게 된 계기였다고 생각했습니다.