CS307 Project2: Unix Shell Programming & LKM for task information

Junjie Wang 517021910093

May 23, 2019

1 Programming Thoughts

1.1 Part 1: Unix Shell Programming

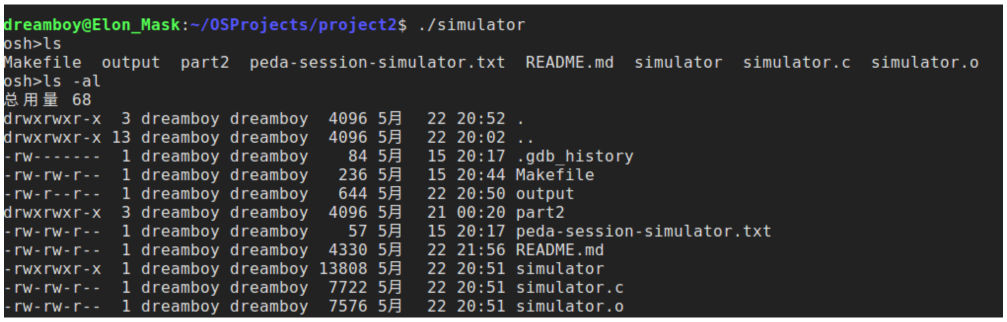
The main logic is to use a while loop to get inputs from the user. Then create a child process(using $fork()$) to execute the command.
To support the history logic, we simply need to maintain one more variable($char*last\_input$). And when receving $'!!'$, copy $last\_input$ into the $input$ string.
To support the & logic, we simply need to modify a little bit about the behavior of the parent process(do not wait until the child process exits). Another thing worth attention here is using $signal()$ to prevent the **zombies**.
To support the redirection logic, the idea is to use $dup2()$ function, closing the stdin/stdout file descriptors and reset its pointer according to the file descriptor we are going to open(e.g. output.txt).
To support the pipe($'|'$), the idea is to create the child process of the child process(also using $fork()$). Pipe is acommplished through the IPC between these two processes. $dup()$ function is used here, actually quite similar to the $dup2()$.

1.2 Part 2: Linux Kernel Module For Task Information

The idea is to get the pid from user's input in $proc\_write()$ function. Then use $task\_pid()$ to get corresponding process and output the pid, command and state of this process. Another thing worth attention is that there should be a logic to handle invalid pid(simply return 0);

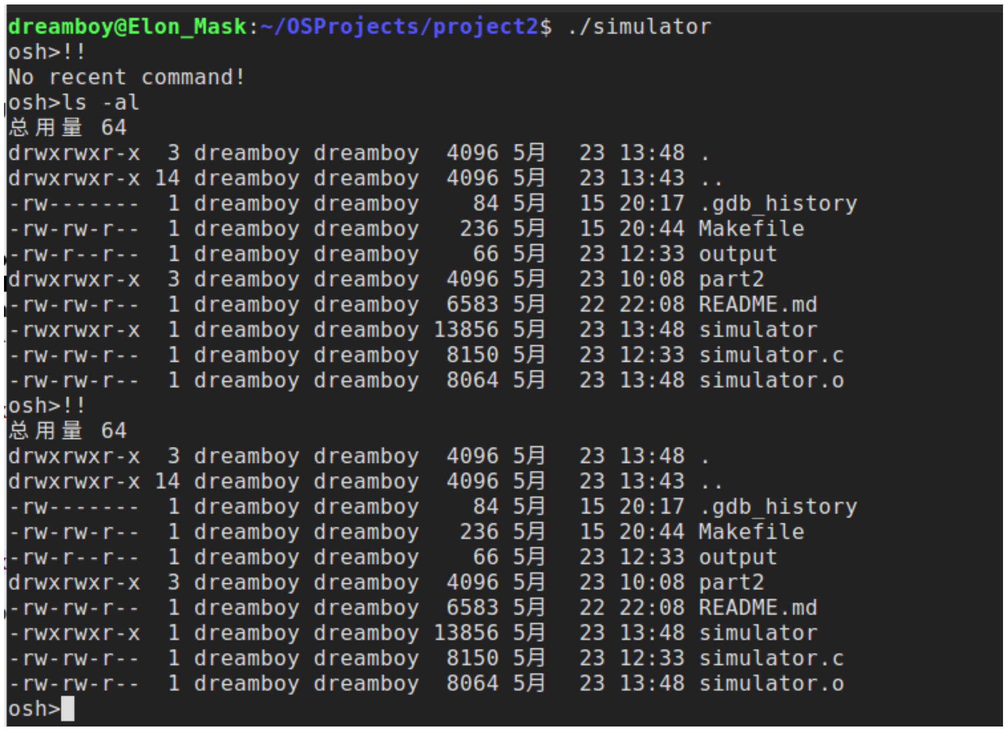
2 Execution Results And Snapshots

Execution results are shown as belows:



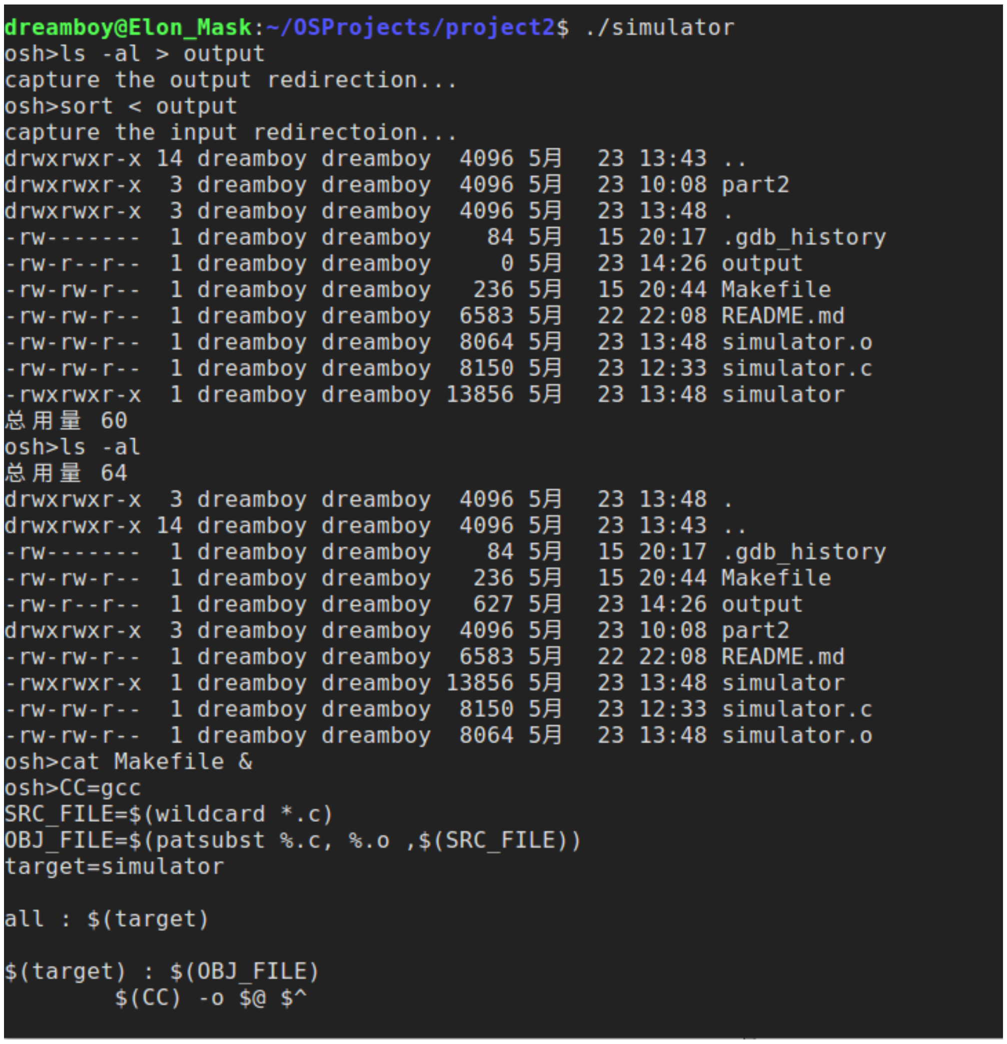
```
dreamboy@Elon_Mask:~/OSProjects/project2$ ./simulator
osh>ls
Makefile  output  part2  peda-session-simulator.txt  README.md  simulator  simulator.c  simulator.o
osh>ls -al
总用量 68
drwxrwxr-x  3 dreamboy dreamboy  4096 5月  22 20:52 .
drwxrwxr-x 13 dreamboy dreamboy  4096 5月  22 20:02 ..
-rw-------  1 dreamboy dreamboy    84 5月  15 20:17 .gdb_history
-rw-rw-r--  1 dreamboy dreamboy   236 5月  15 20:44 Makefile
-rw-r--r--  1 dreamboy dreamboy   644 5月  22 20:50 output
drwxrwxr-x  3 dreamboy dreamboy  4096 5月  21 00:20 part2
-rw-rw-r--  1 dreamboy dreamboy    57 5月  15 20:17 peda-session-simulator.txt
-rw-rw-r--  1 dreamboy dreamboy  4330 5月  22 21:56 README.md
-rwxrwxr-x  1 dreamboy dreamboy 13808 5月  22 20:51 simulator
-rw-rw-r--  1 dreamboy dreamboy  7722 5月  22 20:51 simulator.c
-rw-rw-r--  1 dreamboy dreamboy  7576 5月  22 20:51 simulator.o
```

Figure 1: Primitive Version



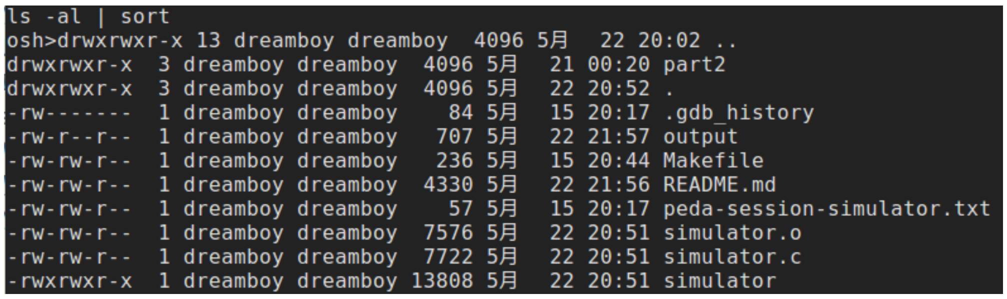
```
dreamboy@Elon_Mask:~/OSProjects/project2$ ./simulator
osh>!!
No recent command!
osh>ls -al
总用量 64
drwxrwxr-x  3 dreamboy dreamboy  4096 5月  23 13:48 .
drwxrwxr-x 14 dreamboy dreamboy  4096 5月  23 13:43 ..
-rw-------  1 dreamboy dreamboy    84 5月  15 20:17 .gdb_history
-rw-rw-r--  1 dreamboy dreamboy   236 5月  15 20:44 Makefile
-rw-r--r--  1 dreamboy dreamboy    66 5月  23 12:33 output
drwxrwxr-x  3 dreamboy dreamboy  4096 5月  23 10:08 part2
-rw-rw-r--  1 dreamboy dreamboy  6583 5月  22 22:08 README.md
-rwxrwxr-x  1 dreamboy dreamboy 13856 5月  23 13:48 simulator
-rw-rw-r--  1 dreamboy dreamboy  8150 5月  23 12:33 simulator.c
-rw-rw-r--  1 dreamboy dreamboy  8064 5月  23 13:48 simulator.o
osh>!!
总用量 64
drwxrwxr-x  3 dreamboy dreamboy  4096 5月  23 13:48 .
drwxrwxr-x 14 dreamboy dreamboy  4096 5月  23 13:43 ..
-rw-------  1 dreamboy dreamboy    84 5月  15 20:17 .gdb_history
-rw-rw-r--  1 dreamboy dreamboy   236 5月  15 20:44 Makefile
-rw-r--r--  1 dreamboy dreamboy    66 5月  23 12:33 output
drwxrwxr-x  3 dreamboy dreamboy  4096 5月  23 10:08 part2
-rw-rw-r--  1 dreamboy dreamboy  6583 5月  22 22:08 README.md
-rwxrwxr-x  1 dreamboy dreamboy 13856 5月  23 13:48 simulator
-rw-rw-r--  1 dreamboy dreamboy  8150 5月  23 12:33 simulator.c
-rw-rw-r--  1 dreamboy dreamboy  8064 5月  23 13:48 simulator.o
osh>
```

Figure 2: Add support for history feature



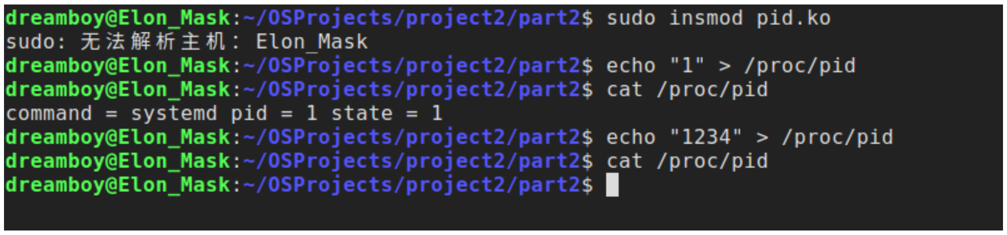
```
dreamboy@Elon_Mask:~/OSProjects/project2$ ./simulator
osh>ls -al > output
capture the output redirection...
osh>sort < output
capture the input redirectoion...
drwxrwxr-x 14 dreamboy dreamboy  4096 5月  23 13:43 ..
drwxrwxr-x  3 dreamboy dreamboy  4096 5月  23 10:08 part2
drwxrwxr-x  3 dreamboy dreamboy  4096 5月  23 13:48 .
-rw-------  1 dreamboy dreamboy    84 5月  15 20:17 .gdb_history
-rw-r--r--  1 dreamboy dreamboy     0 5月  23 14:26 output
-rw-rw-r--  1 dreamboy dreamboy   236 5月  15 20:44 Makefile
-rw-rw-r--  1 dreamboy dreamboy  6583 5月  22 22:08 README.md
-rw-rw-r--  1 dreamboy dreamboy  8064 5月  23 13:48 simulator.o
-rw-rw-r--  1 dreamboy dreamboy  8150 5月  23 12:33 simulator.c
-rwxrwxr-x  1 dreamboy dreamboy 13856 5月  23 13:48 simulator
总用量 60
osh>ls -al
总用量 64
drwxrwxr-x  3 dreamboy dreamboy  4096 5月  23 13:48 .
drwxrwxr-x 14 dreamboy dreamboy  4096 5月  23 13:43 ..
-rw-------  1 dreamboy dreamboy    84 5月  15 20:17 .gdb_history
-rw-rw-r--  1 dreamboy dreamboy   236 5月  15 20:44 Makefile
-rw-r--r--  1 dreamboy dreamboy   627 5月  23 14:26 output
drwxrwxr-x  3 dreamboy dreamboy  4096 5月  23 10:08 part2
-rw-rw-r--  1 dreamboy dreamboy  6583 5月  22 22:08 README.md
-rwxrwxr-x  1 dreamboy dreamboy 13856 5月  23 13:48 simulator
-rw-rw-r--  1 dreamboy dreamboy  8150 5月  23 12:33 simulator.c
-rw-rw-r--  1 dreamboy dreamboy  8064 5月  23 13:48 simulator.o
osh>cat Makefile &
osh>CC=gcc
SRC_FILE=$(wildcard *.c)
OBJ_FILE=$(patsubst %.c, %.o ,$(SRC_FILE))
target=simulator

all : $(target)

$(target) : $(OBJ_FILE)
        $(CC) -o $@ $^
```

Figure 3: Add support for & and the redirection



```
ls -al | sort
osh>drwxrwxr-x 13 dreamboy dreamboy  4096 5月  22 20:02 ..
drwxrwxr-x  3 dreamboy dreamboy  4096 5月  21 00:20 part2
drwxrwxr-x  3 dreamboy dreamboy  4096 5月  22 20:52 .
-rw-------  1 dreamboy dreamboy    84 5月  15 20:17 .gdb_history
-rw-r--r--  1 dreamboy dreamboy   707 5月  22 21:57 output
-rw-rw-r--  1 dreamboy dreamboy   236 5月  15 20:44 Makefile
-rw-rw-r--  1 dreamboy dreamboy  4330 5月  22 21:56 README.md
-rw-rw-r--  1 dreamboy dreamboy    57 5月  15 20:17 peda-session-simulator.txt
-rw-rw-r--  1 dreamboy dreamboy  7576 5月  22 20:51 simulator.o
-rw-rw-r--  1 dreamboy dreamboy  7722 5月  22 20:51 simulator.c
-rwxrwxr-x  1 dreamboy dreamboy 13808 5月  22 20:51 simulator
```

Figure 4: Add support for the pipe



```
dreamboy@Elon_Mask:~/OSProjects/project2/part2$ sudo insmod pid.ko
sudo: 无法解析主机: Elon_Mask
dreamboy@Elon_Mask:~/OSProjects/project2/part2$ echo "1" > /proc/pid
dreamboy@Elon_Mask:~/OSProjects/project2/part2$ cat /proc/pid
command = systemd pid = 1 state = 1
dreamboy@Elon_Mask:~/OSProjects/project2/part2$ echo "1234" > /proc/pid
dreamboy@Elon_Mask:~/OSProjects/project2/part2$ cat /proc/pid
dreamboy@Elon_Mask:~/OSProjects/project2/part2$
```

Figure 5: The part 2, LKM for task information

3 Code Explanation

3.1 part 1

An important part is using the *dup*(), *dup*2() and *execvp*() functions.

```
/* find a minimum value in current available file descriptors and set its
   pointer as the oldfd */
int dup(int oldfd);
/* only a little different from dup(). In dup2() we can specify the newfd
   we want to use. It newfd is already open, close it first */
int dup2(int oldfd, int newfd);
/* exec family functions. Note that the first file parameter will be search
   in $(PATH) environment variable and the last parameter for the */
int execvp(const char *file, const char *argv[]);
```

The logic for the pipe is as follows. For the child process, we first close the stdin and the write port of the pipe. Then use *dup*() function to get the 0 file descriptor and set its pointer to the read port of the pipe. The logic for the parent process is similar.

```
int p[2], ppid;

pipe(p);
ppid = fork();
if(ppid < 0)  {perror("ppid create error!");exit(1);}
else if(ppid == 0){
/* for the child of the child process, set it as the executor of the second
   commmand */
close(0);
close(p[1]);
if(dup(p[0]) < 0) {perror("dup error!");exit(1);}
res = execvp(pipe_args[0], pipe_args);
if(res < 0) { perror("cmd2 execution error!"); exit(1); }
}else{
/* for the parent process, it will execute the first command */
/* close stdout */
close(1);
/* close one port */
close(p[0]);
if(dup(p[1]) < 0) {perror("dup error!");exit(1);}
res = execvp(valid_args[0], valid_args);
if(res < 0) { perror("cmd1 execution error!"); exit(1); }
}
```

3.2 part 2

To get the *command*, *state*, *pid* information, we need to refer to *linux/sched.h* and use logic as follows in *proc_read*() function(Note that *l_pid* is parsed from *proc_write*() function.

```
static ssize_t proc_read(struct file *file, char __user *usr_buf, size_t
    count, loff_t *pos)
{
int rv = 0;
char buffer[BUFFER_SIZE];
static int completed = 0;
struct task_struct *tsk = NULL;

if (completed) {
completed = 0;
return 0;
}

tsk = pid_task(find_vpid(l_pid), PIDTYPE_PID);

completed = 1;

/* return 0 if no this pid */
if( tsk == NULL)  return 0;
else rv = sprintf(buffer, "command = %s pid = %d state = %ld\n", tsk->comm,
    tsk->pid, tsk->state);

// copies the contents of kernel buffer to userspace usr_buf
if (copy_to_user(usr_buf, buffer, rv)) {
rv = -1;
}

return rv;
}
```