

## Assignment 1: Shell

*Due: Sunday, Oct. 6, 2024, 11:59PM*

### 1 Introduction

- The goal of this assignment is to implement a simple shell in xv6-riscv.
- A shell in Linux is a user program that reads command inputs and executes them. If you open a terminal, you should see something similar to the following, followed by a “\$” sign waiting for your input.

```
username@ubuntu $
```

- Try the `ls` command to display a list of files in the `/usr/bin/` directory.

```
username@ubuntu $ ls /usr/bin
alias      awk          bg           bison       bzip2       c++
cc         cpp          curl        env         find        g++
gcc        git          grep        ...
```

- For the command input of `ls /usr/bin/`, the shell creates a child process using `fork()` and makes the child execute the command via `exec()`.

```
int pid = fork();
if(pid > 0) {
    // The parent process (i.e., shell) waits for the child to finish.
    wait(0);
}
else if(pid == 0) {
    // The child process executes the command.
    char *argv[3] = {"ls", "/usr/bin/", 0};
    exec(argv[0], argv);
}
```

- The shell repeats such execution for every user command in a loop until `exit` is entered.

```
username@ubuntu $ exit
```

- There is not only one shell program in Linux but many. Common shells are `bash`, `csh`, `ksh`, `tsh`, `zsh`, etc. You can check the current shell type as follows.

```
username@ubuntu $ echo $SHELL
/bin/bash
```

- Since the shell is a user program, you can make it run another shell as a child process, such as the following.

```
username@ubuntu $ bash
bash $
```

### 2 Implementation

- To start the assignment, go to `xv6-riscv/`, download `shell.sh`, and run the script to update `xv6-riscv`.

```
username@ubuntu $ cd xv6-riscv/
username@ubuntu $ wget https://casl.yonsei.ac.kr/teaching/eee3535/shell.sh
username@ubuntu $ chmod +x shell.sh
username@ubuntu $ ./shell.sh
```

- If the update is successful, you can run a new shell program called `ysh`. The only working commands in the `ysh` skeleton code are `cd` and `exit`, which do not invoke programs but are reserved keywords in the shell.

```
username@ubuntu $ make qemu
```

```
...
```

```
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 1
-nographic -global virtio-mmio.force-legacy=false -drive file=fs.img,if=none,
format=raw,id=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0
```

```
EEE3535 Operating Systems: booting xv6-riscv kernel
EEE3535 Operating Systems: starting sh
$ ysh
EEE3535 Operating Systems: starting ysh
$ exit
EEE3535 Operating Systems: closing ysh
$
```

- The following shows a part of the `ysh` skeleton code in `user/ysh.c`.

```
/* user/ysh.c */

...

int main(int argc, char **argv) {

    ...

    // Read and run input commands.
    while((cmd = readcmd(buf)) && !runcmd(cmd)) ;

    ...

}

// Run a command.
int runcmd(char *cmd) {
    if(!*cmd) { return 0; } // Empty command

    // Skip leading white space(s).
    while(*cmd == ' ') { cmd++; }
    // Remove trailing white space(s).
    for(char *c = cmd+strlen(cmd)-1; *c == ' '; c--) { *c = 0; }

    if(!strcmp(cmd, "exit")) { return 1; } // exit command
    else if(!strncmp(cmd, "cd ", 3)) { // cd command
        if(chdir(cmd+3) < 0) { fprintf(2, "Cannot cd %s\n", cmd+3); }
    }
    else {
        // EEE3535 Operating Systems
        // Assignment 1: Shell
    }
    return 0;
}
```

- The `main()` gets a user command from `readcmd()` and executes it through `runcmd()`. The `readcmd()` function is complete in the skeleton code. Do not modify the `readcmd()` and `main()` functions.
- `runcmd()` first checks if `cmd` is an empty string. Then, it removes leading and trailing white spaces for a non-empty string.

- If the command is `exit`, it returns 1, which will break the while loop in `main()` and end the `ysh` program. If the first three characters of `cmd` are `"cd "`, it will change to the specified directory.
- Otherwise, the command is expected to execute a program. This part of the function is left for the assignment.
- `runcmd()` returns 0 for no errors or a non-zero value for an exit condition (e.g., fork failure).
- You are allowed to add functions or structs to `ysh.c` or introduce additional variables in `runcmd()`. But do not modify the `readcmd()` and `main()` functions.

### 3 Validation

- Every shell has a different parsing scheme for command inputs, so they work differently when complex commands are put together. But simple commands run the same because there cannot be different interpretations.
- This assignment will test if your `ysh` implementation can handle simple commands with pipe (`|`), series (`;`), background (`&`), and/or output redirect (`>`) symbols.

1. **A single command:** The shell forks a child and makes it run a program. The maximum number of args (i.e., length of `argv[]`) that a command can have is 16 in `xv6-riscv` (i.e., `#define max_args 16`).

```
$ ysh
EEE3535 Operating Systems: starting ysh

$ forktest
fork test
fork test OK

$ grep RISC README
but is implemented for a modern RISC-V multiprocessor using ANSI C.
You will need a RISC-V "newlib" tool chain from

$ ls cat echo init kill
cat          2 3 33656
echo         2 4 32536
init         2 7 33000
kill         2 8 32448
```

2. **Two commands with a pipe:** The command on the left side of a pipe forwards its output to the input of the right-side command. There can be two or more child processes depending on the implementation.

```
$ whoami | grep ID
Student ID: 2024143535

$ grep xv6 README | wc
5 54 319
```

3. **Two commands in a series:** The command on the right of a semicolon is executed after the left-side command is done using `wait()`.

```
$ stressfs; wc stressfs0
stressfs starting
write 0
write 1
write 2
write 3
write 4
read
read
read
read
read
0 1 10240 stressfs0
```

- ```
$ grind &
$ forktest
fork test
fork test OK
$ BABABABABABABABABABABABABABABABABABABA ... // grind never ends.
```

- ```
$ cat README > README2
$ cat README2
v6 is a re-implementation of Dennis Ritchie's and Ken Thompson's Unix
Version 6 (v6).  xv6 loosely follows the structure and style of v6,
...

```

- ```
$ ysh
EEE3535 Operating Systems: starting ysh
$ usertests > usertests.out &; stressfs | grep read | wc; cat usertests.out; exit
5 5 25
usertests starting
test copyin: OK
test copyout: OK
test copyinstr1: OK
test copyinstr2: OK
test copyinstr3: OK
test rwsbrk: OK
EEE3535 Operating Systems: closing ysh
$
```

- ```
$ ./tar.sh
$ ls
2024143535.tar  kernel  LICENSE  Makefile  mkfs  README  tar.sh  user
```

- Your teaching assistant (TA) will grade your assignments. If you think your assignment score is incorrect, discuss your concerns with the TA. Always be courteous when contacting the TA. If no agreement is reached between you and the TA, escalate the case to the instructor to review your assignment. Refer to the course website for the contact information of the TA and the instructor: <https://cas1.yonsei.ac.kr/eee3535>
- Arguing for partial credits for no valid reasons will be considered as a cheating attempt and lose the assignment score.