



Rapport Intermédiaire du Projet de Systèmes d'Exploitation Centralisés

« Minishell »

Guohao DAI, Groupe E

Première année - Département Sciences du Numérique

2020-2021

Question 1 (Lancement d'une commande) and Question 2 (Exemple) :

When starting the minishell, commands will be read in an infinite loop. In addition to internal instructions, the startup of other commands is done through `fork()`, a child process created.

As for the parent process, if the child process is in the foreground, wait for it to end, otherwise we execute it in the background.

```
void shell_loop(void)
{
    while (1) {
        init(); //Initialization
        if (read_command() == -1){
            break;
        }
        parse_command();
        printf_command();
        execute_command();
    }
}
```

Question 3 (Enchaînement séquentiel des commandes) :

If the parent process reads the next command without waiting for the end of the child process, the display of the command prompt may appear before the child process is executed.

Question 4 (Commandes internes) :

The processing of internal commands is to check whether the command is not an internal command in `parse_command()` in `shell_loop`. If it is, it will be processed by the corresponding function in `builtin.c`. If not, it will be processed by `execute_command()`.

```
int parse_command(void)
{
    /*
     *If the first command is Enter
     *Return 0
     */
    if(check("\n"))
        return 0;

    /*
     * Determine whether it is an internal
     command,
     * If it is, just execute it
     * Internal command : "cd" "exit" "jobs"
     "stop" "pg" "fg"
     */
    if(builtin())
        return 0;

    /* 1.Parse the first command */
    get_command(0);

    /* 2.Determine whether there is input
    redirection */
    if(check("<"))
        get_name(infile);

    /* 3.Determine if there is a pipe */
    int i;
    for(i=1; i<PIPELINE; i++) {
        if(check("|"))
            get_command(i);
        else
            break;
    }

    /* 4.Determine whether there is output
    redirection */
    if(check(">")) {
        if(check(">")) {
            append = 1;
        }
        get_name(outfile);
    }
}
```

<pre> /* 5.Determine whether a background job */ if(check("&")) backnd = 1; /* 6.Determine whether the command is over '\n' */ if(check("\n")) { </pre>	<pre> cmd_count = i; return cmd_count; } else { fprintf(stderr, "Command line syntax error!\n"); return -1; } return 0; } </pre>
--	--

Question 5 (Lancement de commandes en tâche de fond) :

The “readcmd.c” provided and the “parse.c” I wrote have a “background” attribute, which indicates whether to start the process in the background. In my file, when the flag backnd = 1, we add the child process to the active process list, and the parent process does not have to wait for its end.

Question 6 (Gérer les processus lancés depuis le shell) :

The jobs command displays the status of jobs that have been started in the current shell environment. If the JobID parameter does not specify a specific job, the status information of all active jobs is displayed. If the termination of a job is reported, the shell deletes the job's process ID from the list known by the current shell environment.

“bg” moves the process to the background to run (Background);

“fg” moves the process to the foreground to run (Foreground).

Question 8 (Redirections) :

- > Redirect output to a file or device overwrite the original file
- >! The output is redirected to a file or device to force overwrite the original file
- >> Redirect the output to a file or device append the original file
- < Input redirection to a program

The redirection function in the shell is realized through the operation of “dup/dup2” on standard input and standard output.

“dup” selects the one with the smallest value from the unused file descriptors, and copies the pointer to the file in the file descriptor pointed to by the oldfd parameter to the selected file descriptor, that is, the two new and old file descriptors point to the same open the file, two file descriptors share the file offset (position), flag and lock.

“dup2” can specify the value of the new file descriptor through the parameter newfd, and copy the file pointer of the file descriptor pointed to by the oldfd parameter to the specified file descriptor (so when newfd already points to the file, it needs to be closed and released first), the old and new file descriptors point to the same file.

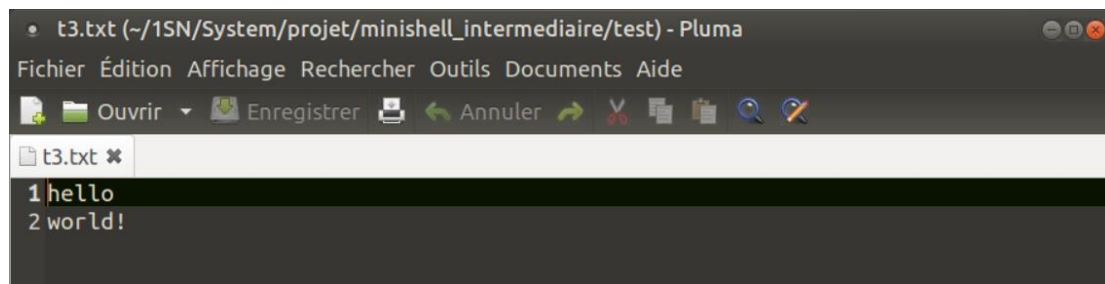
For example:

I typed “hello ” in t1.txt, and typed “world!” in t2.txt. Then enter “cat t1.txt t2.txt >> t3.txt” in the minishell. Observe the content in t3.txt

```

Terminal
Eichier Édition Affichage Recherche Terminal Aide
gdai@roucool:~/1SN/System/projet/minishell_intermediaire$ ./minishell
[Guohao@ /home/gdai/1SN/System/projet/minishell_intermediaire]$ ls
builtin.c def.h execute.h init.c main.c parse.c README.txt
builtin.h execute.c extern.h init.h minishell parse.h test
[Guohao@ /home/gdai/1SN/System/projet/minishell_intermediaire]$ cd test
[Guohao@ /home/gdai/1SN/System/projet/minishell_intermediaire/test]$ cat t1.txt t2.txt >> t3.txt
[Guohao@ /home/gdai/1SN/System/projet/minishell_intermediaire/test]$

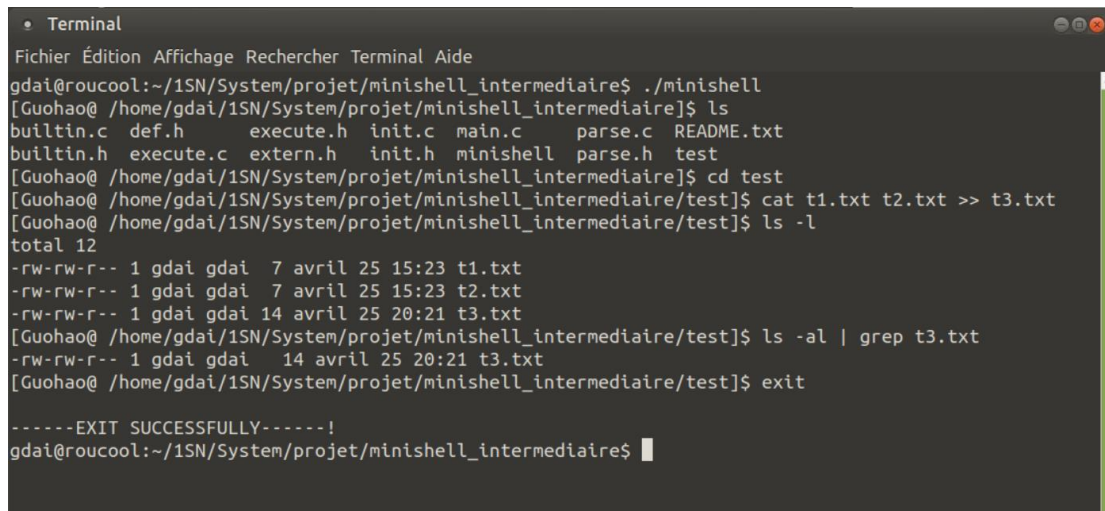
```



```
t3.txt (~/.1SN/System/projet/minishell_intermediaire/test) - Pluma
Fichier Édition Affichage Rechercher Outils Documents Aide
Ouvrir Enregistrer Annuler
t3.txt x
1 hello
2 world!
```

Question 9 (Tubes simples) and Question 10 (Pipelines) :

Briefly, the redirection operator “>” Connect command file together with the file receiving command output; the pipe symbol “|” connect command up command, a second command is received by the output of the first command.



```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
gdai@roucool:~/1SN/System/projet/minishell_intermediaire$ ./minishell
[Guohao@ /home/gdai/1SN/System/projet/minishell_intermediaire]$ ls
builtin.c def.h execute.h init.c main.c parse.c README.txt
builtin.h execute.c extern.h init.h minishell parse.h test
[Guohao@ /home/gdai/1SN/System/projet/minishell_intermediaire]$ cd test
[Guohao@ /home/gdai/1SN/System/projet/minishell_intermediaire/test]$ cat t1.txt t2.txt >> t3.txt
[Guohao@ /home/gdai/1SN/System/projet/minishell_intermediaire/test]$ ls -l
total 12
-rw-rw-r-- 1 gdai gdai 7 avril 25 15:23 t1.txt
-rw-rw-r-- 1 gdai gdai 7 avril 25 15:23 t2.txt
-rw-rw-r-- 1 gdai gdai 14 avril 25 20:21 t3.txt
[Guohao@ /home/gdai/1SN/System/projet/minishell_intermediaire/test]$ ls -al | grep t3.txt
-rw-rw-r-- 1 gdai gdai 14 avril 25 20:21 t3.txt
[Guohao@ /home/gdai/1SN/System/projet/minishell_intermediaire/test]$ exit
-----EXIT SUCCESSFULLY-----!
gdai@roucool:~/1SN/System/projet/minishell_intermediaire$
```