Myshell - Documentation

## Overview

The “myshell” program is a C language program that acts as an interactive shell program which the user can input UNIX command and execute it inside “myshell”.

The basic flow of the program is to first wait for the user’s input and examine the command line. By saving the command as an array of string, the shell program can check whether the command is needed to be run in background or include pipe data flow. The shell program can show the information of process – timeX information, including the process ID, name, real time, user time and system time.

# Process layering

In the program, most of the time, the structure of process will be 3-layered, 4-layered if pipe is involved.

Wait till the child process terminated

Print timeX information if needed

Wait till the grandchild process terminated

If run in background

Parent Process

Grandchild Process

Child

Process

Run the command through execvp()

fork()

fork()

After putting the command line into an array of string, the parent will fork a child process. The child process will then immediately fork one more grandchild to run the command. If pipe is needed while running the command, the grandchild process will fork more grand-grandchild like normal pipe program.

The reason of using 3 layered structure is to let the job of checking process statistics done by the child process while the parent process can decide whether wait for the whole process including printing timeX information or just left it in the background.

# timeX

“timeX” is a built-in command to see the process statistic. In this program, the child process will go into a while loop and keep updating the variables until the grandchild process becomes a zombie process. The RTIME (real time) will be obtained by comparing the starting time of process and the ending time of the process. If the indicator “isTimeX” is true, the statistics will be printed out before the child process returned.

# Background process

If a ‘&’ is in the end of the command line, the indicator “background” will be set. At this situation, the parent process will not wait (execute a waitpid() function) for the child process’s termination. The user can immediately input the next command while the child process is running by its own.

# Pipe

The “myshell” program support pipe while the user likes to execute commands like “ ls | wc”, accept at most 5 commands i.e. 4 pipes. The program will count the number of pipe needed while handling the command line input. If pipe is needed, the program will prepare 4 pipe and start creating multiple child processes. Each child processes contain the pipe action and the command that has been divided into parts. All the processes will be run in grandchild of the program.

# Signal

The program has installed the signal handler for SIGINT. In different situations, the program will behave differently. When the user try to send the SIGINT signal in the command line input stage, the signal will be caught by the handler and clear the current input of command. After that, the program will then back to running. As the command line has been cleared, the program will skip one loop and then back to the initial stage of command input.

When a foreground process is running, the parent program and child program‘s signal handler of SIGIGN will be set into SIG\_IGN. Therefore, only the reaction from grandchild process which is running the command will be valid.

When a process is running in the background, the child process will block the SIGIGN signal so that the grandchild process will not be affected.

Also, a signal handler for SIGCHLD is installed. The program will obtain the variable of process ID from the child process and then print out the “Done” statement.