

IMPLEMENT A JOB SUBMISSION PROGRAM

2023-24 PROGRAMMING #1

TASK

You are going to implement a C program that acts as **an interactive job submission program**, named **JCshell**.

OBJECTIVES

An assessment task related to ILO4 [Practicability]

- Have hands-on practice in designing and developing a shell program, which involves the **creation, management, and coordination** of multiple processes

A learning activity related to ILO 2a

To learn how to use various important Unix system functions

- to perform **process creation** and program execution
- to support **interaction between processes** by using signals and pipes
- to get the processes' running statistics by **reading the /proc file system**

INTERACTIVE SHELL

The program accepts a single command or a job that consists of a sequence of commands linked together with pipes (|) and executes the corresponding command(s) with the given arguments

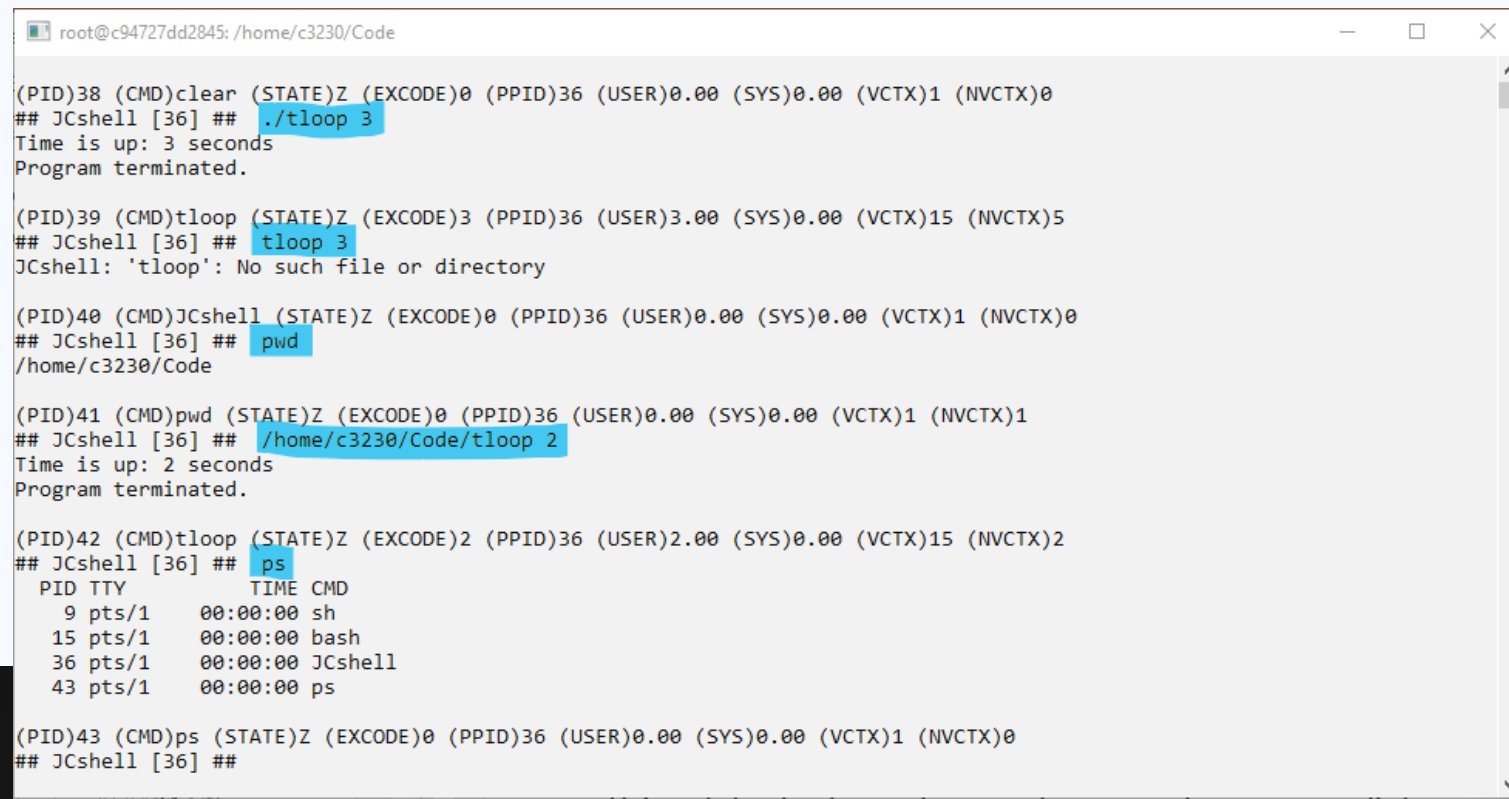
- JCshell waits for all child processes to terminate before accept next user's request

```
root@c94727dd2845: /home/c3230/Code
root@c94727dd2845: /home/c3230/Code# ./JCshell
## JCshell [36] ##
## JCshell [36] ##
## JCshell [36] ##
## JCshell [36] ##
## JCshell [36] ##
## JCshell [36] ##
## JCshell [36] ## ps
  PID TTY          TIME CMD
   9 pts/1    00:00:00 sh
  15 pts/1    00:00:00 bash
  36 pts/1    00:00:00 JCshell
  37 pts/1    00:00:00 ps
(PID)37 (CMD)ps (STATE)Z (EXCODE)0 (PPID)36 (USER)0.00 (SYS)0.00 (VCTX)1 (NVCTX)0
## JCshell [36] ##
```

```
root@c94727dd2845: /home/c3230/Code
root@c94727dd2845: /home/c3230/Code# ./JCshell
## JCshell [30] ## ps f
  PID TTY          STAT TIME  COMMAND
   9 pts/1        Ss   0:00  /bin/sh
  15 pts/1        S    0:00  \_ bash
  30 pts/1        S+   0:00  \_ ./JCshell
  31 pts/1        R+   0:00  \_ ps f
   1 pts/0        Ss+  0:00  bash
(PID)31 (CMD)ps (STATE)Z (EXCODE)0 (PPID)30 (USER)0.00 (SYS)0.00 (VCTX)1 (NVCTX)0
## JCshell [30] ## ls
JCshell JCshell.c Makefile cpu-mechanisms.txt forever forever.c segfault segfault.c tloop tloop.c
(PID)32 (CMD)ls (STATE)Z (EXCODE)0 (PPID)30 (USER)0.00 (SYS)0.00 (VCTX)8 (NVCTX)1
## JCshell [30] ## ./tloop 5
Time is up: 5 seconds
Program terminated.
(PID)33 (CMD)tloop (STATE)Z (EXCODE)5 (PPID)30 (USER)4.98 (SYS)0.01 (VCTX)15 (NVCTX)8
## JCshell [30] ## ps
  PID TTY          TIME CMD
   9 pts/1    00:00:00 sh
  15 pts/1    00:00:00 bash
  30 pts/1    00:00:00 JCshell
  34 pts/1    00:00:00 ps
(PID)34 (CMD)ps (STATE)Z (EXCODE)0 (PPID)30 (USER)0.00 (SYS)0.00 (VCTX)1 (NVCTX)1
## JCshell [30] ##
```

FEATURE – LOCATE AND EXECUTE ANY VALID COMMAND

It is able to locate and execute any valid program (i.e. programs that are already compiled) by giving an absolute path (starting with /) or a relative path (starting with ./) or by searching directories under the \$PATH environment variable.



```
root@c94727dd2845: /home/c3230/Code

(PID)38 (CMD)clear (STATE)Z (EXCODE)0 (PPID)36 (USER)0.00 (SYS)0.00 (VCTX)1 (NVCTX)0
## JCshell [36] ## ./tloop 3
Time is up: 3 seconds
Program terminated.

(PID)39 (CMD)tloop (STATE)Z (EXCODE)3 (PPID)36 (USER)3.00 (SYS)0.00 (VCTX)15 (NVCTX)5
## JCshell [36] ## tloop 3
JCshell: 'tloop': No such file or directory

(PID)40 (CMD)JCshell (STATE)Z (EXCODE)0 (PPID)36 (USER)0.00 (SYS)0.00 (VCTX)1 (NVCTX)0
## JCshell [36] ## pwd
/home/c3230/Code

(PID)41 (CMD)pwd (STATE)Z (EXCODE)0 (PPID)36 (USER)0.00 (SYS)0.00 (VCTX)1 (NVCTX)1
## JCshell [36] ## /home/c3230/Code/tloop 2
Time is up: 2 seconds
Program terminated.

(PID)42 (CMD)tloop (STATE)Z (EXCODE)2 (PPID)36 (USER)2.00 (SYS)0.00 (VCTX)15 (NVCTX)2
## JCshell [36] ## ps
  PID TTY          TIME CMD
   9 pts/1    00:00:00 sh
  15 pts/1    00:00:00 bash
  36 pts/1    00:00:00 JCshell
  43 pts/1    00:00:00 ps

(PID)43 (CMD)ps (STATE)Z (EXCODE)0 (PPID)36 (USER)0.00 (SYS)0.00 (VCTX)1 (NVCTX)0
## JCshell [36] ##
```

FEATURE – PRINTING THE RUNNING STATISTICS

All information about the statistics of a process can be found in the stat and status files under the `/proc/{pid}` directory

PID	The process ID of the terminated process
CMD	The filename of the command – obtain from <code>/proc/{pid}/stat</code> or <code>/proc/{pid}/status</code>
STATE	The process state (for our application, it is always in Zombie (Z) state) – obtain from <code>/proc/{pid}/stat</code> or <code>/proc/{pid}/status</code>
EXCODE	The exit code of the terminated process – obtain from <code>/proc/{pid}/stat</code> or from <code>waitpid()</code>
EXSIG	The name of the signal that caused the process to terminate
PPID	The process's parent ID – obtain from <code>/proc/{pid}/stat</code> or <code>/proc/{pid}/status</code>
USER	The amount of time (in seconds) the process was in user mode – obtain from <code>/proc/{pid}/stat</code>
SYS	The amount of time (in seconds) the process was in kernel mode – obtain from <code>/proc/{pid}/stat</code>
VCTX	The number of voluntary context switches experienced by the process – obtain from <code>/proc/{pid}/status</code>
NVCTX	The number of non-voluntary context switches experienced by the process – obtain from <code>/proc/{pid}/status</code>

FEATURE - | PIPE

It supports the **|** (pipe) operator between commands.

Assume no more than 4 pipes (i.e., 5 commands)

```
root@c94727dd2845: /home/c3230/Code

(PID)44 (CMD)clear (STATE)Z (EXCODE)0 (PPID)36 (USER)0.00 (SYS)0.00 (VCTX)1 (NVCTX)0
## JCshell [36] ## cat cpu-mechanisms.txt | grep process | grep trap | wc -l
3
(PID)45 (CMD)cat (STATE)Z (EXCODE)0 (PPID)36 (USER)0.00 (SYS)0.00 (VCTX)12 (NVCTX)0
(PID)46 (CMD)grep (STATE)Z (EXCODE)0 (PPID)36 (USER)0.00 (SYS)0.00 (VCTX)4 (NVCTX)0
(PID)47 (CMD)grep (STATE)Z (EXCODE)0 (PPID)36 (USER)0.00 (SYS)0.00 (VCTX)5 (NVCTX)0
(PID)48 (CMD)wc (STATE)Z (EXCODE)0 (PPID)36 (USER)0.00 (SYS)0.00 (VCTX)4 (NVCTX)0
## JCshell [36] ## ps
  PID TTY          TIME CMD
   9 pts/1    00:00:00 sh
  15 pts/1    00:00:00 bash
  36 pts/1    00:00:00 JCshell
  49 pts/1    00:00:00 ps

(PID)49 (CMD)ps (STATE)Z (EXCODE)0 (PPID)36 (USER)0.00 (SYS)0.00 (VCTX)1 (NVCTX)0
## JCshell [36] ## cat cpu-mechanisms.txt | | grep trap | wc -l
JCshell: should not have two | symbols without in-between command
## JCshell [36] ##
```

FEATURE - | PIPE

JCshell always waits for all commands to complete before printing the running statistics of all terminated child process(es).

The order of the output should reflect the termination order

```
root@c94727dd2845: /home/c3230/Code
root@c94727dd2845:/home/c3230/Code# ./JCshell
## JCshell [76] ## ./tloop 10 | ./tloop 5 | ./tloop 2
Time is up: 2 seconds
Program terminated.
(PID)79 (CMD)tloop (STATE)Z (EXCODE)2 (PPID)76 (USER)2.00 (SYS)0.00 (VCTX)10 (NVCTX)2
(PID)78 (CMD)tloop (STATE)Z (EXSIG)Broken pipe (PPID)76 (USER)5.00 (SYS)0.00 (VCTX)11 (NVCTX)3
(PID)77 (CMD)tloop (STATE)Z (EXSIG)Broken pipe (PPID)76 (USER)10.00 (SYS)0.00 (VCTX)14 (NVCTX)13
## JCshell [76] ##
```


FEATURE - EXIT

Built-in command: *exit* - to terminate the JCshell program

```
root@c94727dd2845: /home/c3230/Code
root@c94727dd2845:/home/c3230/Code# ./JCshell
## JCshell [51] ## ps f
  PID TTY          STAT       TIME COMMAND
    9 pts/1        Ss          0:00 /bin/sh
   15 pts/1        S           0:00  \_ bash
   51 pts/1        S+          0:00  \_ ./JCshell
   52 pts/1        R+          0:00      \_ ps f
    1 pts/0        Ss+         0:00 bash

(PID)52 (CMD)ps (STATE)Z (EXCODE)0 (PPID)51 (USER)0.00 (SYS)0.00 (VCTX)1 (NVCTX)0
## JCshell [51] ## exit all
JCshell: "exit" with other arguments!!!
## JCshell [51] ## ps
  PID TTY          TIME CMD
    9 pts/1        00:00:00 sh
   15 pts/1        00:00:00 bash
   51 pts/1        00:00:00 JCshell
   53 pts/1        00:00:00 ps

(PID)53 (CMD)ps (STATE)Z (EXCODE)0 (PPID)51 (USER)0.00 (SYS)0.00 (VCTX)1 (NVCTX)0
## JCshell [51] ## exit
JCshell: Terminated
root@c94727dd2845:/home/c3230/Code# ps
  PID TTY          TIME CMD
    9 pts/1        00:00:00 sh
   15 pts/1        00:00:00 bash
   54 pts/1        00:00:00 ps
root@c94727dd2845:/home/c3230/Code#
```

FEATURE – SIGINT SIGNAL HANDLING

JCshell should not be terminated by SIGINT (Ctrl-c) signal.

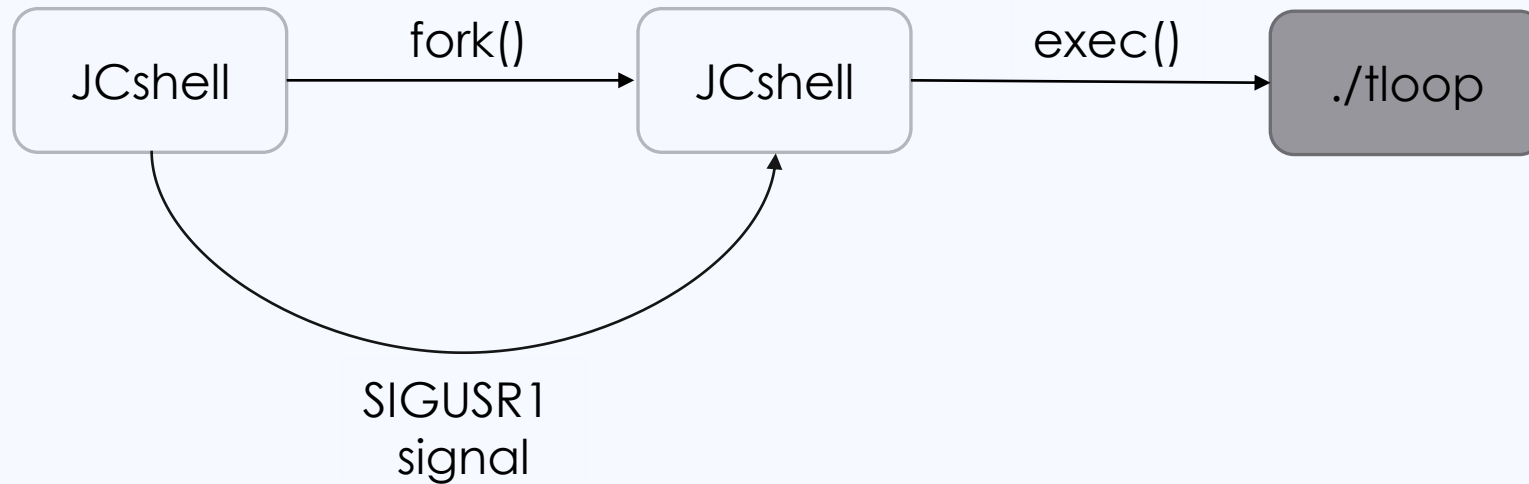
Child process should response to SIGINT signal if the signal is detected; whether the it will terminate or not should depend on the defined logic in that child program

If a process is terminated by signal, displays the signal information

```
root@c94727dd2845: /home/c3230/Code
root@c94727dd2845: /home/c3230/Code# ./JCshell
## JCshell [56] ## ^C
## JCshell [56] ## ^C
## JCshell [56] ## ^C
## JCshell [56] ## ps
PID TTY TIME CMD
 9 pts/1 00:00:00 sh
15 pts/1 00:00:00 bash
56 pts/1 00:00:00 JCshell
57 pts/1 00:00:00 ps
(PID)57 (CMD)ps (STATE)Z (EXCODE)0 (PPID)56 (USER)0.00 (SYS)0.00 (VCTX)1 (NVCTX)0
## JCshell [56] ## ./forever
^CReceives SIGINT!! IGNORE IT :)
Receives SIGINT!! IGNORE IT :)
^C
(PID)58 (CMD)forever (STATE)Z (EXSIG)Terminated (PPID)56 (USER)56.65 (SYS)0.00 (VCTX)16 (NVCTX)61
## JCshell [56] ## ./tloop 10
^C
(PID)69 (CMD)tloop (STATE)Z (EXSIG)Interrupt (PPID)56 (USER)2.95 (SYS)0.00 (VCTX)15 (NVCTX)10
## JCshell [56] ## ./tloop 100
(PID)70 (CMD)tloop (STATE)Z (EXSIG)Killed (PPID)56 (USER)27.47 (SYS)0.00 (VCTX)15 (NVCTX)28
## JCshell [56] ## ./segfault
(PID)74 (CMD)segfault (STATE)Z (EXSIG)Segmentation fault (PPID)56 (USER)0.49 (SYS)0.00 (VCTX)16 (NVCTX)1
## JCshell [56] ##
```

FEATURE – SIGUSR1 SIGNAL

Uses SIGUSR1 signal for controlling when the child process starts running the target command



COMPUTER PLATFORM TO USE

You are expected to develop your program

- Using Windows WSL2 with Ubuntu app; Or
- Using Ubuntu docker image; Or
- Using a native Ubuntu installation on your notebook/laptop; Or
- Using the Dept's Linux server *workbench2.cs.hku.hk*

After fully tested locally, upload the program to the workbench2 for the final tests

Your programs must be written in C and successfully compiled with gcc

- Make sure that your program can be compiled **without any errors**

SUBDIVIDE THE TASK INTO THREE STAGES

Stage 1 – support one command (with arguments) only

- parse the command line
- create new process to execute a valid command
- wait for child process to terminate and get its exit status
- print terminated child process's statistics

Lab 1

Stage 2

- handle SIGINT signal correctly
- use SIGUSR1 signal for control
- implement the exit command

Stage 3 - support pipes

- parse the command line to get more than one commands separate by '|' sign
- use pipe() and dup2() system calls to set up a pipe between processes
- wait for child processes to terminate and get their exit status

} Lab 2

SOME UTILITY PROGRAMS

Utility.zip

- Contains some useful C programs for testing certain features of the shell program

`forever.c`
`segfault.c`
`tloop.c`

SUBMISSION OF ASSIGNMENT

You should name your program JCshell_StudentNumber.c

Submit your programs to the course's Moodle web site

Add your signature in the header of your submitted program and include clear comments

```
/******  
* Student name and No.: Harry Potter 3015999999  
* Development platform: Ubuntu docker container  
* Remark: Complete all features  
******/
```

GRADING RUBRIC

For the details, please read project specification document

Documentation	Student's info (-0.5)
	Sufficient comments (-1)
Correctness of the program (11 points)	Process creation and execution (3 points)
	Process creation and execution – use of ' ' (3 points)
	Print statistics (2.5 points)
	Use of signals (2 points)
	Built-in command: exit (0.5 points)