Goals

By completing this assignment, you should know how to develop a program in Linux platform and to make system calls related to process management.

Introduction

You need to implement a C++ program called `BP` that allows the user to run multiple processes at the background. While processes are running at the background, the user can input command to display information of the background processes, stop, restart or kill a background process.

Requirements

1. Your BP needs to show a prompt for user input as follows.

```
cs3103-01:/home/lec/vlee> BP
BP >
```

2. `BP` accepts the following commands from the user and takes the corresponding action.

```
BP >bg [name of executable file] [a list of arguments]
```

Action: `BP` runs the executable file with a list of arguments at the background and continues to accept input from the user.
Example: `BP` runs the executable file `demo1` with a list of [arguments]: `running 2 5` at the background.

```
BP >bg demo1 running 2 5
```

```
BP >bglist
```

Action: Display the process id and name of all non-terminated background processes.
Example:

```
BP >bglist
16529: demo1
16605: demo2
```

```
BP >bgstop [pid]
```

Action: Stop the process with process id `pid` and display a message.
Example:

```
BP >bgstop 16529
16529 stopped
```

```
BP >bgrestart [pid]
```

Action: Resume the execution of the stopped process with process id `pid` and display a message.
Example:

```
BP >bgrestart 16529
16529 restarted
```

```
BP >bgstate [pid]
```

Action: Display the process id and state (running, stopped or terminated) of the process with process id `pid`.
Example:

```
BP >bgstate 16529
16529: running
```

```
BP >bgkill [pid]
```
Action: Terminate the process with process id `pid` and display a message.
Example:
```
BP >bgkill 16529
16529 killed
```

```
BP >exit
```
Action: `BP` executes `bgkill` to terminate all background processes, if any, and exits.
Example:
```
BP >exit
16605 killed
16607 killed
cs3103-01:/home/lec/vlee>
```

3. BP should display a message after a background process has completed.
   Example:
   ```
   16529 completed
   ```

4. `BP` should display a short and clear message in any other cases such as invalid commands, non-existing process id, no background process, etc.

Hints

- You may use `fork()` and `execvp()` so that the parent process accepts user input and the child process executes the background process.
- When you use `fork()`, it is important that you do not create a *fork bomb*, which easily eats up all the resources allocated to you. If this happens, you can try to use the command "`kill`" to terminate your processes (http://cslab.cs.cityu.edu.hk/supports/unix-startup-guide). However, if you cannot log into your account any more, you need to ask CSLab for help to kill your processes.
- You may use `waitpid()` with an option `WNOHANG` to check if a background process has completed.
- You may use the `kill()` system call to send a signal to a process, e.g., a `SIGTERM` signal to terminate a process.
- You should study the man pages of the system calls before calling them in your program. For example, the following command displays the man pages of `kill()` in Section 2.
  ```
  cs3103-01:/home/lec/vlee> man 2 kill
  ```

Helper programs

demo.cpp

- This demo program can be used to act as a background process for testing your `BP` as its execution can be visualized by displaying a word every few seconds a number of times.
- This program takes three arguments, `word`, `interval` and `times`.
- The first argument `word` is a single word to be displayed repeatedly.
- The second argument `interval` is the number of seconds between two consecutive displays of the word.
- The third optional argument `times` is the number of times the word to be displayed. If this argument is missing, the word will be displayed infinite number of times. CAUTION: DROPPING THIS ARGUMENT MAY CAUSE A FORK BOMB.
- For example, the following command displays the word "running" 5 times in 2-second interval.
  ```
  cs3103-01:/home/lec/vlee> demo running 2 5
  ```

args.cpp

- This example program shows how to display a list of all arguments passed to it.
- To compile the program, use the following command.
  ```
  cs3103-01:/home/lec/vlee> g++ args.cpp -lreadline -lhistory -o args
  ```

Marking

- You program will be tested on our CSLab Linux servers (cs3103-01, cs3103-02, cs3103-03). **If an executable file cannot be generated and running successfully on our Linux servers, it will be considered as unsuccessful.**
- Marking scheme (total: 100%):

o `bg`	20%
o `bglist`	10%
o `bgstop`	10%
o `bgrestart`	10%
o `bgstate`	10%
o `bgkill`	10%
o `exit`	10%
o Handling of other cases	10%
o programming style and in-program comments	10%

Submission

- This assignment has to be done individually or by a group of two students. You are encouraged to discuss the high-level design of your solution with your classmates but you **must** implement the program on your own. Academic dishonesty such as copying another student's work or allowing another student to copy your work, is regarded as a serious academic offence.
- Each submission consists of two files: a source program file (.cpp file) and a text file containing all possible outputs produced by your program (.txt file).
- Use your student ID(s) to name your submitted files, such as 5xxxxxxx.cpp, 5xxxxxxx.txt for individual submission, or 5xxxxxxx_5yyyyyyy.cpp, 5xxxxxxx_5yyyyyyy.txt for group submission. Only ONE submission is required for each group.
- Submit the files to Canvas.
- The deadline is **10:00 a.m., 23-FEB-18 (Friday)**. No late submission will be accepted.

Questions?

- Contact Miss LIANG, Yu at yliang22-c@my.cityu.edu.hk or your course lecturer.