Version Saturday, October 28th 2017

Week 06

Deadline : Saturday, November 4th 2017 15:00

Fork

1. In this tutorial, we will learn how fork process works in OS with some C program
2. Login to your badak account
3. Change your directory to "work" and create new directory named “work06” inside “work” directory

   ```
   $ cd work
   $ mkdir work06
   ```

4. Move all the files that required to your work06 directory (file “materi-lab.zip” on SCELE).

   The files you need to move:

   (1) 04-fork.c

   (2) 05-fork.c

   (3) 06-fork.c

   (4) 10-fork.c

   (5) 11-fork.c

   (6) cascafork.c

   (7) cascafork2.c

   (8) Makefile

5. Change your directory to “work06” directory
6. Make a new lab06.txt file

   ```
   $ vi lab06.txt
   ```

7. Fill the first row of file lab06.txt with your github username like: (without the hashtag)

   ```
   # Github Account: [YOUR_GITHUB_USERNAME]
   ```

8. Compile the file in the work06 directory

   ```
   $ make
   ```

Learning Fork

1. Take a look about fork manual in linux:

   ```
   $ man fork
   ```

2. Run the 04-fork program

   ```
   $ ./04-fork
   ```

3. Learn the code from the 04-fork .c program and learn the output from the program
4. Do the rest for file 05-fork, 05-fork.c, 06-fork, 06-fork.c, 10-fork, 10-fork.c, 11-fork, 11-fork.c

5. Then you must answer some of question below and write it on lab06.txt point number 1:
   a) Explain what each program you have just ran does! Write in format like: (without the hashtag)

```
# 04-fork.c: [YOUR_EXPLAINATION]
# 05-fork.c: [YOUR_EXPLAINATION]
# 06-fork.c: [YOUR_EXPLAINATION]
# 10-fork.c: [YOUR_EXPLAINATION]
# 11-fork.c: [YOUR_EXPLAINATION]
```

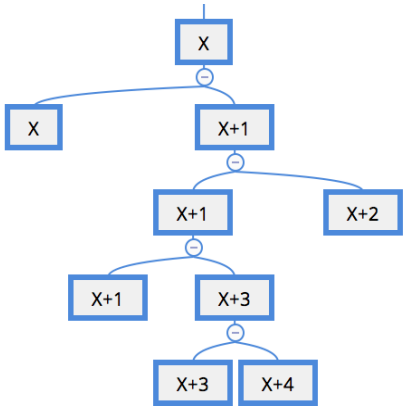
   b) What is PID, PPID, getpid() and getppid() function?
   c) What is the return value of fork() method and what does the return value mean?
   d) Explain sleep(), wait(), and waitpid() function with each parameter means!
   e) Write your knowledge about fork! (don't forget to add the sources if you use reference!).
   f) Every time you execute a C program that prints PID, why does it not always print the same output? For example, when you print a PID of 06520, it will print PID of 06535 the next time you execute the same program.

Code a Fork

1. Run the cascafork program

```
$ ./cascafork
```

2. Learn the code from the cascafork.c program and learn the output from the program
3. In your directory there is a C file named cascafork2.c . Edit the file to have the fork process like the fork tree. Your objective is to **edit the program until have an output like the example** (the PID and PPID number might be different from the example depends on the number of processes run on your OS, but the sequence of the output should be the same) Fork tree: (using the same notation just like what you have learned in class, and X defines relative PID)



Output example:

```
This is PID[07023]
This is PID[07025]
This is PID[07024]
This is PID[07027]
This is PID[07026]
```

hint:

- you can re-compile the program using `$ make` command for every change you make
- just change the first if() bracket and within only (line 11 through line 22), do not change the rest

4. After completed the program, do not forget to compile the program and add your output program to result.txt

```
$ make
$ ./cascafork2 > result.txt
```

5. Then you must answer some of question bellow and write it on lab06.txt point number 2:
   - What does the `(pid_t)` do?
   - Why is the `waitpid(-1,NULL,0);` function executed TWICE? (hint: find out what will happen if the function is only executed ONCE).

Privacy Matters, Encryption and Digital Signature using GnuPG

1. Hash and sign your works so the other knows it is truly your works

```
$ sha1sum * > SHA1SUM
$ sha1sum -c SHA1SUM
$ gpg --sign --armor --detach SHA1SUM
```

2. Verify the works

```
$ gpg --verify SHA1SUM.asc
```

3. Create a tar ball. Tar is a way to create an archive file. You can ask uncle G for more information

```
$ cd ..
$ tar cvfj work06.tbj work06/
```

4. Encrypt the tar file

```
$ gpg --output work06.tbj.gpg --encrypt --recipient OSTEAM --recipient
your@email.com work06.tbj
```

*Use the same email as your Email input on GnuPG key generator.

5. Copy the file to your github account, under the file week06/

```
$ cp work06.tbj.gpg ~/os172/week06/work06.tbj.gpg
```

6. Change your directory to **os172/week06/**
7. Remove file named "`dummy`"
8. Check whether there is a file named "`work06.tbj.gpg`" if you don't find it, do the copy once more.
9. Push the change to GitHub server
10. Done.

Review your Work

Don't forget to check your files/folders. After this lab, your current `os172` folder should looks like:

```
os172
	key
		mypublickey1.txt
	log
		log01.txt
		log02.txt
		log03.txt
		log04.txt
		log05.txt
		log06.txt
	SandBox
		<some_random_name>
	week00
		report.txt
	week01
		lab01.txt
		report.txt
		whyStudyOS.txt
		what-time-script.sh
	week02
		work02.tbj.gpg
			*work02
				*00-toc.txt
				*01-public-osteam.txt
				*02-ls-al.txt
				*03-list-keys1.txt
				*04-list-keys2.txt
				*hello.c
				*hello
```

```
                *status.c
                *status
                *loop.c
                *loop
                *exercise.c
                *exercise
                *SHA1SUM
                *SHA1SUM.asc
week03
      work03.tbj.gpg
            *work03
                  *.profile
                  *sudo-explanation.txt
                  *what-is-boot.txt
                  *SHA1SUM
                  *SHA1SUM.asc
week04
      work04.tbj.gpg
            *work04
                  *01-public-osteam.txt
                  *lab04.txt
                  *global-char.c
                  *global-char
                  *local-char.c
                  *local-char
                  *open-close.c
                  *open-close
                  *write.c
                  *write
                  *result1.txt
                  *result2.txt
```

```
                *demo-file1.txt
                *demo-file2.txt
                *demo-file3.txt
                *demo-file5.txt
                *00-pointer-basic.c
                *00-step-1
                *00-step-2
                *00-step-3
                *00-step-4
                *SHA1SUM
                *SHA1SUM.asc
week05
      Work05.tbj.gpg
            *vm-to-memory.c
            *vm-to-memory
week06
      work06.tbj.gpg
            *04-fork.c
            *04-fork
            *05-fork.c
            *05-fork
            *06-fork.c
            *06-fork
            *10-fork.c
            *10-fork
            *11-fork.c
            *11-fork
            *cascafork.c
            *cascafork
            *cascafork2.c
            *cascafork2
            *Makefile
            *result.txt
            *lab06.txt
            *SHA1SUM
            *SHA1SUM.asc
```

```
week07
	dummy
week08
	dummy
week09
	dummy
week10
	dummy
xtra
	dummy
```

keep in mind for every files/folders with wrong name, you will get penalty point.

*means file that should be inside the archived file.