

# ASSIGNMENT 6: PROCESS CONTROL

CS 3424 - Systems Programming

Sam Silvestro - UTSA

## IMPORTANT NOTICE ABOUT THIS ASSIGNMENT

**Warning:** do **NOT** utilize the fox machines for performing or testing this assignment! You can and **WILL** cause a `fork()`-bomb, causing the system(s) to become unstable and unusable.

Instead, you **MUST** utilize servers `hen04.cs.utsarr.net` or `hen03.cs.utsarr.net`. Note that these servers are accessible only in the following ways:

- (1) Within the UTSA network (i.e., using the on-campus network);
- (2) via VDI from home; or,
- (3) via connecting to a fox machine *first*, *then* proceeding to SSH into one of these two hen machines.

If either server becomes unstable or unusable due to a student's out-of-control project, email me to alert me to the situation and I will have the system(s) reset.

For this assignment, you will use **C**'s process control functions to exercise basic process creation. Your program should have the following functionality:

### Concurrency

Given **up to six** commands separated by commas (a comma will be its own token, with whitespace around it) and **any number of** arguments each, execute all commands *concurrently* (i.e., in parallel and not one after the other).

Each process (excluding the parent) should print their PID followed by their PPID followed by their command without arguments **to stderr**. Any normal behavior by the command issued should remain the same. You do not need to account for redirection or pipelining.

Your program **should not produce orphans**. Toward verification of this, if a process's PPID is 1, it is an orphan that has been adopted by the `init` process. In solving this problem, be sure that your child processes still run concurrently. Do not wait for one process to complete before starting a new one.

### Example

```
$ assign6 ls -a , pwd , cat hello.txt
PID: 35003, PPID: 35002, CMD: ls
PID: 35004, PPID: 35002, CMD: pwd
PID: 35005, PPID: 35002, CMD: cat
/home/ssilvestro
```

```
. .. courses Desktop Downloads  
this is the contents of hello.txt
```

Note that since the processes are happening in parallel, the order of the output is not guaranteed.

## Compiling Your Program

Your submission will include a makefile so the following command can be used to compile your code.

```
$ make assign6
```

This should produce an executable file named **assign6**. For more information about the make utility, check the related document on Blackboard.

If you attempt the extra credit, `$ make assign6-ec` should compile the extra credit portion to **assign6-ec**.

## Extra Credit (25 points)

Create a second program, **assign6-ec** which takes **exactly two** commands, with **any number of** arguments each, and separated by a comma. The program should pipe the output of the first command to the second using `dup2()`.

### Example

```
$ assign6-ec ls -l , sort -k5 -n
```

should behave as

```
$ ls -l | sort -k5 -n
```

Extra credit is not given to late assignments. All requirements must be met for both parts of the assignment to qualify for extra credit.

## Assignment Data

This assignment does not include sample input files since it will only take arguments.

## Program Files

Your submission should consist of up to five files:

- **assign6.c** - the main file which is compiled (**required**)
- **assign6.h** - an optional header file if necessary
- **assign6-ec.c** - the main file which is compiled for extra credit (**required for extra credit**)
- **assign6-ec.h** - an optional header file if necessary

- **Makefile** - the makefile to make the **assign6** executable (**required**).

## Submission

Turn your assignment in via Blackboard. Your zip file, named **a8-abc123.zip** should contain only the files described above.

If you attempt the extra credit, name your file **a8-abc123\_EC.zip**. Without the **\_EC**, your submission will be graded as normal.