# COP4600.001C11: Operating Systems
# Project One: Processes and Threads

May 23, 2011

## 1 Instructions

Write a program that requires a single integer argument, call it $n$, taken from the command line, and does the following:

1. The program must invoke fork() to create a single child process.
2. The child process must invoke execv() to replace itself with a program that calculates the $n^{th}$ Fibonacci number.
3. The parent process must pass the argument, $n$, to the child process, then wait for it to complete.
4. Once the child process finishes execution, the parent process must then spawn a new thread that performs the same calculation using the same argument.
5. Again, the parent process must wait for this thread to finish its calculation before continuing.

You must do error and boundary checking so that your program exits gracefully if it is not passed a single integer argument between 1 and 1000. Please note that your program must fork a child process and create a new thread; they are different actions and require different operating system calls. Please use the POSIX threading library, `pthread`.

# 2 Conventions

Your submission will receive severe penalties if it does not adhere to the following conventions.

## 2.1 The Fibonacci Sequence

We will go by the convention that our sequence index starts at 1, and
$f(1) = 0$
$f(2) = 1$
$\forall x > 2 : f(x) = f(x-1) + f(x-2)$

## 2.2 Required Files

Please submit a zip file named `lastname-firstname-proj1.zip` containing the following files.

```
makefile
main.c
fib.c
```

The file `main.c` should contain your program's source code, including the code for a recursive function that your thread uses. The file `fib.c` should itself be independently compilable.

## 2.3 Requirements of the makefile

Your makefile should support the following commands (remember that you do not type the CLI prompt indicator $), when run from the directory containing your source files:

```
$ make all
$ make clean
```

The `all` command should produce your executable file named `proj1`, and the executable program `fib` for use in the child process. The `clean` command should remove all temporary and compiled files, leaving only the source files required of you for this project.

## 2.4 Output Format

Please see the following sample output to see how to structure your printf() format strings. Test commands similar to these will be run after executing `make all`. Your program's output does not include the first two spaces on each of the following lines, which are shown just for clarity. To distinguish the parent process's output from that of the children, both the child process and the child thread should print two leading spaces before any other characters.

Each process must begin by announcing its process ID, and each thread must begin by announcing its thread ID. Your program should also announce when a child process or thread on which it waits has completed. If the user does not supply your program with a valid argument, simply print `incorrect arguments`, and quit.

```
$ ./proj1 3
  pid: 101
    pid: 102
    f(3) = 1
  child finished executing
    tid: 498
    f(3) = 1
  thread finished executing

$ ./proj1 -101
  incorrect arguments

$ ./proj1 1703
  incorrect arguments

$ ./proj1 "hello"
  incorrect arguments

$ ./proj1 5 19
  incorrect arguments
```