ECE 362 – Embedded Operating Systems

Assignment 2

Due: Jan 29th in D2L dropbox

Please read announcements and discussion postings on D2L for assignment requirement updates before submitting

Solutions should only use system calls for obtaining input and generating output. See below the types of system calls you can use. Variations of these calls are OK. If you have questions, please send me an email. Any debugging code can use C library code (e.g. printf, scanf).

Deliverables: Submit separate files in your dropbox: your readme file, makefile and each source code file.

1. **end**

   Write a utility program that takes one or two arguments (*filename* and *n*) and writes to `stdout` the last *n* lines of the file *filename*.
   - If the filename does not exist, the program should display an error.
   - If n is not greater than 0 and less than 10, the program should generate an error
   - If *n* is not specified, the program should write only the last 5 lines.

   Your program should read the file 10 characters at a time.

2. **capture**

   This solution to this problem will require using some system calls that will be discussed in week 3.

   This program should accept the name of a file that includes a list of commands (one per line) to execute. Each line of the file should be executed by a subprocess and the output written to the file `capture.txt`

   For example, given the command file:

```
/bin/ls -l
/bin/cat apple banana
/usr/bin/wc -l -w canteloupe
```

   The output saved to `capture.txt` might be:

```
-rw-------+ 1 schubert them    51 Jan 18 12:07 apple
-rw-------+ 1 schubert them    27 Jan 18 12:09 banana
-rw-------+ 1 schubert them 13179 Jan 18 12:15 cantaloupe
first line of file apple
second line of file apple
banana first and only line
   41    217 cantaloupe
```

`unlink`	remove a file
`open`	open a file for reading/writing
`close`	close a previously open file
`dup`	duplicate an existing file descriptor
`read`	extract info from a file
`write`	put info into a file
`lseek`	move to specified location in a file

`fork`	create a new almost-copy of the current program
`execl`	replace program being executed with a new one
`wait`	wait for child process to complete
`exit`	terminate self
`dup, dup2`	duplicate an existing file descriptor
`kill`	Send a signal to a process

- To send a signal: `kill(pid, sig);`
- To handle a signal: `signal(sig, handler);`