Launch-Tube

### Due Date: September 21, 2014 @ midnight Submission Subject: “Launch Tube”

## General Submission Criteria:

* See Lab 0 for the General Submission Criteria!
* Make a directory in your repository: lab2
* Include all of your Lab2 work within the lab2 directory

## Overview:

In this lab, you will develop two programs. The two programs are designed to be down in two parts. The first part is a program called “launch”, whereas the second part is a program called “tube”. You will also need to modify the original makefile to allow the Professor to build your two software programs.

## Executable Names:

**launch**: a program that allows one to set in motion another program, while also report some basic information.

**tube**: a program that sets to programs into motion that are connected via a tube that transports inter-process communication, while also report some basic information.

## Makefile Targets:

all: (default)

launch:

tube:

clean:

## Part 1 Description: Launch

In this software project, you are to write a command-line tool that launches another process and then have this process execute a particular program. The general steps that this program undergoes include:

* The program forks a child process (see fork(2))
* The parent process prints the PID of the child on stderr
* The child process executes the supplied command (see execve(2))
  + the child needs to prepare the new argv structure
* The parent prints the return value of the child on stderr (see waitpid(2))

## Usage Example:

|  |
| --- |
| $ launch /bin/ls  /bin/ls: $$ = 26843  a.out foo.c  /bin/ls: $? = 0  $  $ launch /bin/ls -l 2>/dev/null  total 12  -rwxr-xr-x 1 steve users 6749 Sep 11 12:10 a.out  -rw-r--r-- 1 steve users 83 Sep 11 12:10 foo.c  $  $ launch /bin/ls -l foobar >/dev/null  /bin/ls: $$ = 26843  /bin/ls: cannot access foobar: No such file or directory  /bin/ls: $? = 2  $ |

## Part 2 Description: tube

In this software project, you are to write a command-line tool that launches two child processes, and sets up a pipe between them for inter-process communication. The general steps that this program undergoes include:

* The program allocates a pipe (see pipe(2))
* The program forks two children
* The parent process prints the PID of both children on stderr (see fprintf(3))
* The parent process closes access to the pipe and the child processes wires the pipe to allow for inter-process communication (see dup2(2))
  + via the standard stdout-stdin approach (see close(2))
* The first process executes the first command, and the second process executes the second command (see execve(2))
  + for the first iteration keep it simple — i.e., the child has no command line args
  + enhance your program to allow for an arbitrary number of command line args
* The program prints the return value of the first child and then the second child on stderr

## Usage Example:

|  |
| --- |
| $ tube /bin/cat filename , /usr/bin/head -2  /bin/cat: $$ = 26843  /bin/head: $$ = 26844  This is the first line  This is the second line  /bin/cat: $? = 0  /bin/head: $? = 0  $  $ tube /bin/cat filename , /usr/bin/head -2 2>/dev/null  This is the first line  This is the second line  $  $ tube /bin/cat filename , /usr/bin/head -2 >/dev/null  /bin/cat: $$ = 26843  /bin/head: $$ = 26844  /bin/cat $? = 0  /bin/head: $? = 0  $ |

## See Also:

* Section 3.6.3 & Figure 3.9 of the Textbook
* man bash (Redirection Section)
* man 2 execve
* man 2 pipe
* <http://proquest.safaribooksonline.com.libproxy.csun.edu/book/operating-systems-and-server-administration/unix/0596009658>