CST 334 (Operating Systems)

Dr. Glenn Bruns

# Lab: C Process API

Try to answer the first couple of problems from memory.

1. What function does a C program call to create a new process?
2. Explain what the function 'exec' does.
3. Answer question 1 in Chapter 5 of OSTEP (our text):

*Write a program that calls fork(). Before calling fork(), have the main process access a variable (e.g., x) and set its value to something (e.g., 100). What value is the variable in the child process? What happens to the variable when both the child and parent change the value of x?*

Save your code, and put a comment in the code that answers the question. You can use the code shown at the bottom of this lab as your starting point.

<http://pages.cs.wisc.edu/~remzi/OSTEP/>

1. Answer question 2 in Chapter 5.

*Write a program that opens a file (with the open() system call) and then calls fork() to create a new process. Can both the child and parent access the file descriptor returned by open()? What happens when they are writing to the file concurrently, i.e., at the same time?*

In my code, the file open call looks something like this:

fd = open("/home/CLASSES/brunsglenn/ctests/proc\_api/temp.c", O\_WRONLY);

Look at the man page for fopen for details, and to see if you need to include anything. Use command man 2 write to get info about the write system call.

By the way, what do you think 'fd' stands for?

1. Answer question 3 in Chapter 5:

*Write another program using fork(). The child process should print “hello”; the parent process should print “goodbye”. You should try to ensure that the child process always prints first; can you do this without calling wait() in the parent?*

(code for problem 3:)

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

// what does a child process see of a variable set by its parent?

int x = 1;

int rc = fork();

if (<your code here>) {

printf("fork error\n");

exit(1);

}

if (<your code here>) {

<your code here>

} else {

<your code here>

}

}