COURSE Final Exam 05

DATE

Your Name Here: __________________________

Instructions: You have TIME for this exam. Please write your answers on the pages in this exam booklet. No scrap paper or additional sheets will be accepted. Watch your time and be concise. Write clearly (illegible answers will be ‘silently ignored’), and *always* check the return value of a system call. Good luck.

prob	points	got	section
1	4		
2	4		
3	4		
4	4		
5	4		
6	4		
7	4		
8	4		
9	5		
10	5		
11	5		
12	5		
13	4		
14	4		
15	4		
16	4		
17	4		
18	4		
a	5		
b	4		
c	6		
d	4		
e	5		

Problems 1-8: Short answer questions. Answer each question clearly, precisely, and refer to specific system calls when appropriate. Write your answer in the space provided.

1. How can you tell when you have reached the end of a file?

2. What is the quickest way to move to the end of a file?

3. How can a parent process find out how its children died?

4. Why is the name of the system call `kill()` a misnomer?

5. What is the difference between `execv()` and `execvp()`?

6. What is an *atomic* operation? Name two.

7. All system calls return -1 on error and not -1 if successful. Name two system calls that never return if they succeed.

8. What is `errno` and what is its purpose?

Part Two

Four problems, each worth 5 points

Problems 9-12: Compare and contrast. Each of these problems mentions two related concepts, system calls, or operations. For each pair, explain briefly and clearly (a) what they have in common, (b) when you would use the first item, and (c) when you would use the second item.

9. `raw mode` vs `non-blocking mode`

10. `pipe` vs `fifo`

11. `fopen()` vs `fdopen()`

12. `EOF on a tty` vs `EOF on a socket`

Someone described the essence of Unix as "The interaction of files and processes." Files store data, and processes operate on data. How similar are files and processes, really? In this problem, you will compare attributes and operations of processes and files.

13. How is a file created? How is a process created?

14. How is a file destroyed/deleted? How is a process destroyed/deleted?

15. How is a file copied? How is process copied?

16. How can one determine the owner of a file? How can a process determine its owner?

17. What is the unique identification number of a file and how does one determine it? What is the unique identification number of a process and how does a process determine it?

18. A file has a size. What does that mean? A process has a size. What does that mean? Can these sizes change?

17. A new Unix utility: a server switchboard

Introduction We have seen how a gopher server sleeps waiting for connections from gopher clients on remote machines. The gopher server accepts calls from clients, receives some request and returns some data. We have seen in class how a time server accepts connections from clients and returns the time of day over a socket. We have also seen how a dictionary server accepts a connection, inputs some text, and outputs a definition from a database.

A Problem This server stuff is great, but soon, a machine is filled with servers sleeping as they wait for calls from clients on other machines. Most of the time, these servers are doing nothing, but they do use slots in the process table and they must be relaunched if they are modified.

A Solution Replace all these separate servers with one program that accepts a request for a server then runs that program for the client. For example, a gopher program might want the gopher service on a given machine. Gopher servers typically are attached to port 70. In this solution, the client calls the machine at a specific port (say port 1200) and asks for service 70. A switchboard operator at port 1200 then runs a gopher server and connects it to the caller's socket. If the client asks for the time of day server, by asking for service 13, the server switchboard runs the time of day program with its output attached to the socket.

For this part of the exam, answer the following questions about the design and components of the server switchboard (hereafter known as servd). Use the space on the remaining pages.

a) What system calls will you need to use in writing **servd**? What role will each play in the program? [5]

b) The set of services known to **servd** can be stored in a file. That file will contain the service number and the program to run. For example, the time-of-day service has number 13 and program `/bin/date`. Explain how your program will use such a file. [4]

c) When **servd** receives a request for a service, it has to run the server program with its input and output attached to the socket. Describe how this connection is accomplished. [6]

d) Write a pseudo-code (or top-level function in C) that outlines the major operation of your program. [4]

e) Where can errors occur in your program? How will you handle them? [6]