csci-e215 Final Exam

May 13, 41996, USA

Your Name Here: ______________________________

Instructions: You have two hours for this exam. This exam is ‘open notes and open book’. 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	7		
b	4		
c	5		
d	4		
e	4		

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. Describe one method of determining the number of bytes in a file.

2. Name and explain three things that would cause the `open()` system call to return an error.

3. Name and explain two reasons why a process would receive a signal.

4. What is the difference between a `pipe` and a `fifo`?

5. What is the role of the `fork()` system call? Explain the meaning of its return value and how the return value is used.

6. What is the meaning and value of 'buffered' input and output?

7. What is the meaning of `echo` mode, and why is it useful?

8. What is `perror()` 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. `program` vs `process`

10. `signal` vs `pipe`

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

12. `wait()` vs `sleep()`

The Internet is a world-wide system of computers that exchange data in a fairly consistent manner. Many of the data-exchange systems were devised and/or based on Unix machines. In this section, you will examine some important aspects of data communication.

13. Sets of information are often stored in files. How does a process gain access to a file of data?

14. A process that has access to information is often called a *server*. Explain, in brief terms, how a server is implemented on a Unix system.

15. What is the purpose of the *curses* library. What role does it play in providing information to Internet users?

16. What is a directory and how can it be used to organize information?

17. What is the role of a *socket* in making information available to interested users?

18. What is the purpose of the *tty driver* on a Unix machine? What role does it play in creating an interface between users and information?

17. A new Unix utility: timeout

Introduction Computer security in a world of interconnected machines is an issue of ever-increasing importance. If you leave your terminal to refill your coffee cup or talk to someone, your connection to the world is left unattended. A simple, and useful, solution is to have your login session terminated if you do not press any keys for a specified amount of time. For example, if you do not press any keys or see any output for three minutes, the login session could be terminated. When you returned from the coffee break, you could login again.

Discussion There are two ways to implement an 'auto-logout' system. In one model, you could write each program (the text-editor, the compiler, the mail reader, the web browser..) to check for idle periods and exit.

In another model, a separate process watches your tty and sees if the tty device was untouched for a specified period. If the terminal was idle for that period, this monitor process would kill all the processes connected to that tty and, thereby, log you out. This method, a single program, clearly is easier.

Problem For this part of the exam, answer the following questions that explore the details of writing a timeout program.

Useful Information The special file called `/dev/tty` is the name of the file that corresponds to the terminal line through which you are connected to the system. Each time you press a key, the file `/dev/tty` is modified, and the last-modified-time is updated to reflect that change. Each time a character is sent to the screen, the file `/dev/tty` is modified and the last-modified-time is updated to reflect the change.

Use the space on the remaining pages.

a) What general algorithm will you use to implement the `timeout` program? [7]

b) When `timeout` determines you have been idle for the specified interval, it will kill all processes attached to your terminal. How can a program determine which processes are attached to your terminal? [4]

c) What system calls will you need to use in writing `timeout`? What role will each play in the program? [5]

d) What arguments will the timeout program take, and how will the program use these arguments? [4]

e) On some machines, for example the one at Harvard called fas, there can be as many as 400 users logged in at once. If each of those 400 users ran a private copy of `timeout`, the system would need to support a lot of redundant processes. Suggest a solution that eliminates the need for so many copies of the same program. [4]