COURSE Final Exam 12

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	6		
10	6		
11	6		
12	6		
13	6		
14	12		
15			
a	6		
b	6		
16			
a	5		
b	5		
c	2		
d	2		

.

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. What is a *zombie* process? What causes them? Why are they a problem?

2. What is the *lowest available file descriptor* principle? Why is it useful?

3. What is an inode table? Name one advantage and one disadvantage of using a large inode table?

4. What is meant by *canonical mode*? Why is it useful?

5. Why do servers create child processes?

6. Files that represent terminals support the `open` and `close` system calls but do not support the `lseek` system call. Why not?

.

7. All system calls return -1 on error and other values on success. Name two system calls that never return if they succeed.

8. The manual and the header files are both useful references for programmers. When would you use man pages, and when would you use header files?

Part Two

Five problems, each worth 6 points

Problems 9-13: *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. `datagram socket` vs `stream socket`

10. `accept()` vs `read()`

11. `dup2()` vs `link()`

12. `utmp` vs `wtmp`

13. `kill()` vs `exit()`

.

'*Enhancing Programs*' Many of the projects for the course have involved building on programs from class. These three problems are based on projects you did for homework.

14. *rm -r* For homework you wrote *pfind*, a program that operates on a tree of files and directories. For this problem you will write a similar program that operates on a tree of files and directories, *deltree*. Write a complete C program called `deltree.c` that removes files and directories in and below a specified directory. The program takes as a single argument the name of a directory and deletes all files and directories below that directory, the contents of that directory, and the directory itself.

Your C program should be complete, ready to compile. Comment lightly, check error codes from system calls. You may not call external programs to do this work. For example, `system("chdir x; rm -r .")` is not acceptable.

.

15. An enhancement to pong: save your job and your car

The pong game can be transfixing. You might lose track of time and forget to put money in the parking meter. Worse, your boss might come in and find you playing games on company time; you might lose your job. Two useful enhancements to the pong game are a parking meter timer and a 'boss key'.

a) *parking meter minder*: The first enhancement is to change the program so you may press the **p** key to tell the game when you just put money in the meter and want to be reminded after two hours pass. The duration can be a command line option.

How would you change the program to add this feature? You cannot just set an alarm timer because pong already uses the timer to keep the ball moving. Describe the changes to the input part of the program, the way you will schedule the reminder, and the way you will notify the player.

.

b) *Boss Key*: You might not want your boss to find you playing pong on company time, so you decide to add a special keystroke to the game that causes the program to clear the screen and run the **pine** email program. When the user exits from **pine**, the pong game resumes where it left off.

How would you change the program to add this feature? Be sure to explain what happens to the timer ticks that drive the ball and what happens to the parking meter minder. You want to be sure to get to the parking meter on time.

16. *Web Server Stats*

Your web server is a big improvement over the original simple web server, but adding page hit statistics makes it even more useful in a large installation. Which pages get lots of hits? Which ones are not popular? Which ones are so popular that requests for them threaten to overload the server or the network capacity? As written, the server receives, as arguments to the GET method, requests for files, directories, and programs.

Your server can answer these questions and manage these problems by keeping track of how many times each page is requested. Furthermore, each page could have an owner and an hourly hit quota. If you are running a web hosting service, you can charge different prices for different page hit quotas. For example, you could charge one rate for up to 20 hits per hour and a higher rate to allow 1000 hits per hour.

By attaching an email address to each page, your server could notify the owner of the page if the hit quota is reached. The customer could then decide to pay for a higher quota.

Finally, the server has to report all the statistics it receives. The site administrator needs a way to ask for these reports, and the program has to be able to produce these reports.

Part I: Data Structures [4 points]

a) What data structures would you use to store these statistics and user information? Why?

Part II: Tabulating Hits [4 points]

b) How do you modify the web server to accumulate these statistics? Where in the program flow do you have to add code? Describe briefly what additions you would make.

Part III: Reports [3 points]

c) How can you implement the server stats report function? How would the site administrator ask for the report? How would the program generate it?

Part IV: Enforcing/Notifying [3 points]

d) How do you enforce hourly hit quotas? How do you notify page owners when their pages hit the quotas?

.

.