Final Exam *Hours left: 2*

csci-

e-215

O

May 17, 1993, 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, be concise, think before you write, look before you leap, listen before you accept, and *always* check the return value of a system call. Good luck.

prob	points	got	section
1	5		
2	5		
3	5		
4	5		
5	5		
6	5		
7	5		
8	6		
9	6		
10	6		
11	6		
12	6		
13	6		
14	6		
1	3		
2	2		
3	5		
4	5		
5	4		
6	4		

Problems 1-7: Each problem describes a particular task. For each one, state which system call(s) you would use to perform that task. In the third column, state clearly and briefly the logic of your solution.

task	system calls	method
Determine if two names are links to the same file.		
Redirect the standard output of a program to a file.		
Turn off character echoing so a user can type a password securely.		
Write a program that exits if Ctrl-C is pressed twice within 3 seconds.		
Get the last five characters in a file.		
Send data to a device like a printer or tape drive.		
Run a program and determine its exit status.		

Problems 8-14: 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.

8. `O_TRUNC` , `O_CREAT`

9. `signal()`, `kill()`

10. `fopen()`, `fdopen()`

11. A child process communicates with its parent by using exit(); a child process communicates with its parent by using a pipe.

12. Stream socket, Datagram socket

13. wait(), pause()

14. blocking input, non-blocking input

15. The Electronic Strip Chart Recorder.

First, a metaphor. Outside of lecture hall C is a strip chart recorder that graphs seismic activity as measured at a geological station somewhere out of the city. There are two parts to the device. The first part is a pen that moves back and forth to indicate the strength of vibrations detected. The second part is the steadily moving roll of paper. The steady motion of the paper allows the 'output' of the pen to be spread out across the paper. The physical separation of the pen output corresponds to the temporal separation of the events.

Next a software example. Imagine there is a program that outputs text at various intervals. It might be a program attached to a device that detects the bright flashes caused by shooting stars and prints messages about their position and intensity. It might be a program that looks for prime numbers and prints out each one it finds. In these examples, the program outputs information in text form to standard output.

Imagine you want to spread the output across a time line, just as the strip chart recorder spreads seismic jiggles across a time line. For example, the prime number strip chart might look like:

```
13:00:00
13:00:10
13:00:20
   1000001819
13:00:30
13:00:40
   1000001857
   1000001887
13:00:50
   1000001917
   1000001927
13:01:00
13:01:10
   1000001957
```

The time markers are printed out every ten seconds. Any output the prime number program produces appears indented by three spaces between the time markers. If this strip-chart program were attached to the shooting star detector, the output would show positions and intensities instead of prime numbers between the time markers.

The program to plot the output of a given program on a time line will be called strip-chart, and it will work as follows:

USAGE

```
strip-chart interval program_name [program_arg1 program_arg2 ..]
```

DESCRIPTION

strip-chart prints out a sequence of time markers separated by a *interval* seconds. A program (and any arguments it might take) is given on the command line. That program is run as a subprogram and its output is combined with the time marker output of **strip-chart**. Each line of output from *program_name* is indented three spaces to distinguish it from the time markers.

If *program_name* closes its standard output (which will occur if the program exit()s, **strip-chart** prints "program exited" on the strip chart and exit.

If **strip-chart** receives an interrupt signal, it prints "interrupted" on the strip chart, shuts down *program_name* and exits.

Discuss a way to implement this program by answering the following questions.

i) Someone suggests a quick solution: write a shell script to spit out the time every n seconds:

```
while true
do
   date ; sleep $INTERVAL
done
```

and run it in the background and run *program_name* in the foreground. On what details of the strip-chart description does this method fail?

ii) Strip-chart has to read standard output and standard error of *program_name* from a pipe and copy the data to strip-chart's own standard output. Explain why *program_name* should not be allowed to write to standard output directly.

iii) Sketch the data connections between the strip-chart program and the *program_name* it runs. Show all pipes, standard input, standard output, and standard error for each process and their connections.

iv) If strip-chart is waiting for input from *program_name*, how can it print the time out every *interval* seconds? (hint: how did pong do this?) Describe what system calls you would use to implement this solution.

v) How does strip-chart handle the interrupt signal? What operations does it have to perform to shut down the child process correctly?

vi) Describe the initialization operations of strip-chart. Describe the main loop of strip-chart.

Answer these questions on this and the following pages.