CSE190 — Final Test

Yee **Spring '00**

Name and ACS Login: Answer Key

There are a total of 16 questions on 9 pages. There are 100 points possible. You may not finish the entire exam — don't be discouraged. Wait until I have passed out exams to everybody before you start. Advice: skim through the entire test to determine which of the problems you can solve quickly and work on those first, rather than getting stuck on a hard problem early and wasting too much of your time on it.

When you can start, you should first make sure that you have all the pages, and write your name and your login name on the first page, and your login name on the top of *all subsequent pages.* Pages of this exam will be separated and graded separately — if you fail to write your name at the top of a page, you will not receive credit for answers on that page. **Write clearly**: if I cannot read your handwriting or your pencil smudges, you will not properly get credit for your answers.

This is a closed book, closed notes exam. You may **not** look at yours or anybody else's books, notes, exam, or otherwise obtain help from another human being, artificial intelligence, metaphysical entity, or space alien. If you have extra sensory perception powers, please curtail their use. If you use ESP or if I see your eyeballs wandering, you will get a zero for the exam. If you must look away from your exam/notes to think, look up at the ceiling / into space or close your eyes.

No electronic computation aids are allowed.

Problem	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	**Total**
Score																	
Possible	5	3	5	5	4	4	6	10	7	5	7	10	12	4	12	1	100

1 (System Calls) For each of the system calls / standard C library calls listed below, write a **Y** next to each one that would be used by a Bourne shell (or descendant) to start the following command running

```
cmda | cmdb
```

and a **N** by each that would not. Do not include system calls or C library routines that would be invoked by either `cmda` or `cmdb` but not the shell.

Y A. `close`
Y B. `dup2`
Y C. `execvp`
N D. `exit`
Y E. `fork`
N F. `mmap`
N G. `open`
N H. `read`
N I. `socket`
N J. `shutdown`

(5pts)

---

2 (Unix Pipelines) When setting up the pipeline

```
cmda | cmdb
```

if the interactive shell that spawns the two subprocesses that will run `cmda` and `cmdb` forget to close the pipe I/O descriptors before invoking `wait` or `waitpid` to wait for the child processes to finish running, what would happen? (Place a check ($\surd$) next to all that applies.)

$\surd$ A. the shell never wakes up from the `wait` or `waitpid`
____ B. the shell dump core
____ C. `cmda` never exits
$\surd$ D. `cmdb` never exits
____ E. `cmda` dumps core
____ F. `cmdb` dumps core
☺ G. the user dumps core

(3pts)

---

3 (Standard C Library) What does the standard C library routine `exit` do?

(5pts)

The `exit` routine performs `atexit` (and `on_exit` for some libc implementations) processing, `writes` out any remaining data in output buffers, attempts to "give back" data read into input buffers that has not been read by the application via a negative offset `lseek`, and then invokes the `exit(2)` system call to terminate the current process.

4 (Kernel Abstractions) Both sockets and pipes permit interprocess communication. In what ways are they different?

(5pts)

Sockets are used to communication over various communications domain with various protocols, e.g., TCP, UDP, IP, PUP, etc. A pipe is a pair of Unix domain sockets, so pipes are a particular instance of the socket abstraction. Sockets are bidirectional; pipes are often unidirectional. Sockets can be used to communicate over a network; pipes cannot. While I/O for both can be initiated with `read` and `write` system calls, data can also be transfered over sockets via the `send`, `sendto`, `sendmsg`, `recv`, `recvfrom`, and `recvmsg` system calls.

5 (Shell Syntax) The following two questions assumes the existance of a program `p`. `p` outputs two data streams; the first to standard output (descriptor 1) and the second to standard error (descriptor 2). You are using either the Bourne shell or bash. (Do *not* use `yaish` command syntax.) You may assume that other than 0, 1, and 2, no other descriptors are in use.

(1) Give the shell command to run `p` so that the outputs that `p` writes to descriptor 1 will be sent to the standard error (descriptor 2) of the invoking environment, and vice versa, i.e., swap the descriptors.

(2) Give the shell command that will capture the standard output data into a file `p.log` and pipe the standard error stream through a paginator program such as `more` or `less`.

(4pts)

```
p 3>&1 1>&2 2>&3 3>&-
p 2>&1 > p.log | more
```

6 (Exit Status Convention) How do Unix processes indicate that they succeeded? How do they indicate that they failed? What are the range of possible exit status values?

(4pts)

An exit status of 0 indicates success. Any non-zero value indicates a failure. Since only 8 bits are available to the parent via the `wait` or `waitpid` system calls, the possible failure values are 1 through 255.

7 (Shell Syntax) Explain what is executed when the following commands run, and what is the exit status of the compound statement.

1: `cmda && cmdb`

2: `cmda || cmdb`

(6pts)

In both cases, the exit status of the compound statement is the exit status of the last command executed.

1: First run `cmda`. If its exit status is success, run `cmdb`; otherwise the compound statement is finished.

2: First run `cmda`. If its exit status is failure, run `cmdb`; otherwise the compound statement is finished.

8 (Unix Networking)

1: Give the sequence of system calls that must be performed by a network client process to talk to a network server.

2: Give the sequence of system calls that must be performed by a server in order to allow clients to talk to it.

(10pts)

1: `socket`, `bind` (optional), `connect` (5 pts)

2: `socket`, `bind`, `listen` (optional), `accept` (5 pts)

9 (Unix Design) The device file `/dev/null` is used for I/O redirection, so that if a process reads from it, an end-of-file indication is immediately received, and if a process writes to it, the written data is simply discarded (ignored).

Some programs such as `grep` takes a `-q` or "quiet" flag to suppress output — the flag is intended for use in scripts, since `grep` exit with success if a match was found. Why do programs like `grep` provide such flags? Can't users of programs like `grep` simply redirect their standard output to `/dev/null`? Explain.

(7pts)

The `-q` flag enables `grep` to exit early, rather than searching through the entire input stream for additional matches that are then ignored because they'll be written out to `/dev/null`.

10 (Perl Basics) List the fundamental data types in `perl`.

(5pts)

`perl` has scalars, which may hold numbers, strings, and references to other objects; arrays; and hashes.

11 (Perl Execution Model) When the system runs a perl program, is the program:

_____ A. interpreted

_____ B. compiled

__√__ C. other

Explain your answer.

(7pts)

Perl programs are first translated into an internal representation, with syntax checks performed on the entire program, and then the internal representation is interpreted. The initial translation step is very similar to compilation, but does not generate machine code.

12 (Script Interpreters) How can a program written in an interpreted language be made into an executable script file that could be invoked directly via the `exec` system call?

(10pts)

The file containing the program should be made executable via `chmod`, and the first line should contain
`#!`*/path/to/interpreter*
to specify the interpreter for the file.

13 (Network Utilities) In order to test your assignment 4 code, you decide to use `netd` to run a server to which you can run the following `yaish` commands:

```
Y$ socket s 7010@host
Y$ cat run.c >| s &
Y$ sclose s 1
Y$ cat <| s
```

which will send the contents of the file `run.c` to the server and display the response. The server should

1: log the data transferred to a file, e.g., `server.$$.log` (the shell variable `$$` expands to the process ID of the shell interpreting the command), and

2: send the logged data back, with each line of text preceded by the string `saw:`$\triangle$. Here $\triangle$ denotes a space character.

Give the appropriate shell command using `netd` and other Unix tools to accomplish this.

(12pts)

```
netd -p 7010 -- sh -c 'cat > server.$$.log; sed "s/^/saw: /" < server.$$.log'
```

or

```
netd -p 7010 -- sh -c 'while read line;
do
        echo "saw: $line";
        echo "$line" >> server.$$.log;
done'
```

14 (Pipes) What happens when a process writes to a descriptor that refers to the input end of pipe and there are no descriptors that refer to the read or output end of the pipe?

(4pts)

The process receives a `SIGPIPE` signal, the default action of which is to terminate the process.

If `SIGPIPE` is explicitly ignored, then the `write` system call will return with `EPIPE`. Most processes do not ignore `SIGPIPE`.

15 (Script Interpreters) Can the interpreter for a script file that is to be run directly by the kernel via the `exec` syscall be itself an interpreted program? If so, how many levels of interpretation is allowed? If not, what happens when you try this on a Solaris machine (e.g., on `ieng9`)?

(12pts)

If an interpreter is to be used via `exec`, it cannot in turn be an interpreted program. If the named interpreter is not an executable binary, the `exec` syscall on the script fails, and the script is typically read by the subshell itself or passed to `sh` (this depends on the shell used to run the script).

16 (Unix History) Is the name "Unix" derived from the word "eunuch"?

(1pt)

(No. This is a freebie.)