October 15, 1991 Name: ______________________

TA: ______________________

CS/CE 218 — Unix and C programming

Test 2

Before you begin, please take a moment to look over the entire test so that you can budget your time.

On this exam, the phrase "shell" refers to the Bourne Shell (or a process running the Bourne Shell). On this test the character printed ` is a "back-quote" (grave accent); the character ' is a normal quote (single quote). All questions refer to the version of HP-UX running on Zippy and Zaphod, which is essentially Unix System V.

English answers will be judged on their completeness and clarity. However, you do *not* need to fill up all the empty space on the page. Conversely, do not feel limited by the space on a page, since you may continue an answer on the back of a page if you wish.

problem	subject	points	score
1	Shell primitives	27	
2	What is `$$`?	3	
3	Mystery program	15	
4	Value of command substitution	10	
5	Give the output	5	
6	Give the output II	5	
7	Trap	4	
8	quotation	20	
9	Write the shell script	11	
Total		100	

This exam (only) counts for 10% of your final grade.

1. (27 points, 3 points each) Write shell code to accomplish each of the following.

   The first three (3) are done for you as examples. (A point will be deducted for each overly complex answer.)

   - Take the contents of the shell's second argument, and make it be the first argument to `cmd1`.
     Answer: `cmd1 $2`
   - Take the shell's first argument, and put its the contents in the file `f`.
     Answer: `echo $1 > f`
   - Take the shell's first argument, and put it in the shell variable `X`.
     Answer: `X=$1`

   (a) Take the shell's first argument, and make its contents be the standard input of `cmd`.

   (b) Take the standard output from `cmd` and put it in the shell variable `X`.

   (c) Take the standard output from `cmd`, and make it the standard input of `cmd2`.

   (d) Take the standard output from `cmd`, and put it in the file `f`.

   (e) Take one line of the shell's standard input, and put it in the shell variable `X`.

   (f) Take the contents of shell variable `X`, and put it out on the shell's standard output.

   (g) Take the entire contents of file `f`, and put it in the shell variable `X`.

   (h) Take the entire contents of file `f`, and make it the first argument to `cmd`.

   (i) Take the exit value of the last command executed by the shell, and put it in the shell variable `X`.

2. (3 points) Briefly describe, in words, what does `$$` mean in a shell script.

3. (15 points) Suppose that the executable file `mystery` contains the following shell script.

```
#!/bin/sh
if test $# -ne 2
then
        echo "message 1"
        exit 5
else
        if test ! -d $1
        then
                echo 'message 2' >&2
                exit 6
        fi
fi
echo `echo $*`
exit 0
```

For each of the following commands, describe any output, which file descriptor (standard output, diagnostic output) the output appears on, and the exit value of the command.

Assume that `dir2` is a directory, and `phantom` *does not exist*.

(a) `mystery`

(b) `mystery phantom dir2`

4. (10 points) Suppose command substitution (i.e., back quotes, such as `` `cmd` ``) was not part of the shell. Briefly describe how it would make shell programming more difficult.

5. (5 points) What is the output of the following shell commands?

```
X=40
cat <<-!
Y=$X
echo Y 10
!
```

6. (5 points) Suppose the executable file `problem` contains the following lines

```
#!/bin/sh
X=$1
```

What output is produced by executing the following commands?

```
X=initial
PATH=.:$PATH
problem foo
echo $X
```

7. (4 points) Show how to use `trap` to ignore hangup signals from within a shell script.

8. (20 points, 2 points each) Give the output of each of the following, assuming that the shell variable x is set to the string y, and that the current working directory contains files named bar and cheese. In your answers, show all blanks as ␣. (Note carefully the difference between the back-quotes ` and the single quotes ' below.)

   (a) echo *

   (b) echo '*'

   (c) echo "*"

   (d) echo `ls`

   (e) echo "`ls`"

   (f) echo '`ls`'

   (g) echo '$x␣␣␣x'

   (h) echo $x␣␣␣x

   (i) echo `echo "$x␣␣␣x"`

   (j) echo `for f in *; do echo $f; done`

9. (11 points) Write a shell script for an interactive `move` command with the following synopsis:

   ```
   move oldfile [newfile]
   ```

   The command must prompt the user for the "newfile" argument if it is missing. If the number of arguments is incorrect, the command should print an error message to its diagnostic output.

   An example of using this command follows, where the user types "file3.new" in response to the prompt:

   ```
   $ ls
   file1.c  file2.txt   file3.c
   $ move file3.c
   New file name? file3.new
   $ ls
   file1.c  file2.txt   file3.new
   $ move file2.txt file3.new file3.c
   Usage: move oldfile [newfile]
   $ echo $?
   1
   ```