Banner number: Name:

# Midterm Exam

## CSCI 2132: Software Development

February 4, 2019

| Marks | |
|---|---|
| Question 1 (6) | |
| Question 2 (10) | |
| Question 3 (6) | |
| Question 4 (9) | |
| Question 5 (9) | |
| Question 6 (5) | |
| Question 7 (5) | |
| **Total (50)** | |

**Instructions:**

- **Write your banner number and name at the top of this page *before continuing to read.***
- **This exam has 10 pages, including this title page. Notify me immediately if your copy has fewer than 10 pages.**
- You are allowed to use **one cheat sheet, letter paper, both sides, 10pt font or larger.**
- Understanding the exam questions is part of the exam. Therefore, **questions will *not* be interpreted.** Proctors will only correct errors in question statements and clarify ambiguities, if any.

**Question 1 (A little bit of everything)** **6 marks**

Mark all of the following statements that are true with a checkmark. No explanation is necessary.

(a) A given file (as identified by its inode) can exist in multiple directories. [✓]

(b) A process is executable code stored in a file. [ ]

(c) For a process to be allowed to read a file with permissions `-w-r----x` (241), its real group ID must match the file's group. [ ]

(d) A soft link (symbolic link) can refer to directories and to files on a different file system (a different disk). [✓]

(e) It is possible to start two processes from the shell so that what one process reads from its `stdin` is whatever the other process writes to its `stdout`. [✓]

(f) Subversion is a distributed version control system. [ ]

**Question 2 (Inodes, paths, file types, shells, and output redirection)** **10 marks**

(a) Consider a file `file1` and a second file `file2` created using the command `ln file1 file2`. Mark all of the following statements that are true with a checkmark:

Deleting `file1` also deletes `file2`. [ ]

`file1` and `file2` refer to the same inode. [✓]

Changing the contents of `file1` also changes the contents of `file2`. [✓]

`file1` and `file2` can have different permissions. [ ]

(b) Assume you execute the following commands with your current working directory being `/home/topstudent`:

```
$ mkdir -p a/b/c a/b/d a/c b/d; cd a/b
```

Mark all of the following statements that are true after this sequence of commands with a checkmark:

The directory `../..` is `/home/topstudent`. [✓]

The directory `.././d` exists. [ ]

The directory `./c/../d` exists. [✓]

The directory `/home/topstudent/b/d` exists and is a subdirectory of `/home/topstudent`. [ ]

(c) bash is a shell that can be used to interact with a Unix system. Mark all of the following statements that are true with a checkmark:

| | |
|---|---|
| bash is part of the Unix kernel. | ☐ |
| bash can be used to start programs. | ✓ |
| bash can be used to automate tasks using scripts. | ✓ |
| bash is one of many shells available on Unix systems. | ✓ |

(d) Which of the following are types of files that exist on a Unix system? Mark each with a checkmark.

| | |
|---|---|
| Regular file | ✓ |
| Block device | ✓ |
| Network card | ☐ |
| Directory | ✓ |

(e) You are testing a new program `prog` you developed. Since you expect it to run for a while, you plan to start it and then go for a coffee. You expect `prog` to produce a lot of normal output on `stdout` and there is a good chance that there will be quite a few error messages sent to `stderr`. In order to be able to inspect these when you come back, you want to capture `stdout` in a file `output` and `stderr` in a file `errors`. Which of the following four commands achieves this:

| | |
|---|---|
| `./prog >> output 2> errors` | ☐ |
| `./prog > output 2>&1 errors` | ☐ |
| `./prog > output 2> errors` | ✓ |
| `./prog stdout=output stderr=errors` | ☐ |

**Question 3 (Users, groups, ownership, and file permissions)** **6 marks**

(a) Mark all of the following statements that hold for a file with permissions `rwxr-x--x`:

- Everybody is allowed to execute the file. ✓
- The file's owner and every member of the file's group can read the file. ✓
- A process whose effective group ID matches the file's owner can write the file. ☐
- Any member of the file's group may change the file's group. ☐

(b) Mark all of the following statements that are true:

- When running a program `prog`, the effective user ID of the resulting process is the same as the owner of the file `prog`. ☐
- `newgrp` allows you to change your effective group to any group you want. ☐
- Every user has a unique user name and a unique user ID. ✓
- Every user can be a member of only one group. ☐

(c) You are a member of the group `qateam` and the owner of a program file `prog`. In order to ensure that only the members of `qateam` are allowed to read and execute (but not write) the file `prog` while not changing your own (the owner's) access permissions, which of the following commands do you use?

```
chown qateam prog; chmod g=rx,o= prog
```
☐

```
chgrp qateam prog; chmod o= prog
```
☐

```
chgrp qateam prog; chmod 750 prog
```
☐

```
chgrp qateam prog; chmod g=rx,o= prog
```
✓

**Question 4 (Wildcards and regular expressions)** **9 marks**

(a) Mark all of the following statements that are true:

Wildcards are a mechanism built into the shell to refer to all files whose names match a given pattern. [✓]

`grep` is a tool that can be used to list all the lines of a text file that match a given wildcard pattern. [ ]

Regular expressions are built into many text editors as a mechanism to search for complex text patterns. [✓]

(b) Provide a command line that uses only `ls` and `grep` to list all files in the current directory whose names consist of three letters (lowercase or uppercase) followed by an arbitrary number of digits (zero digits is possible).

```
ls | grep -E -e '^[A-Za-z]{3}[0-9]*$'
```

(c) Provide a single wildcard expression *expr* so that `ls` *expr* lists the same files as in Question (b) or explain briefly why such a wildcard expression *expr* does not exist.

*This can't be done because we need to check for an arbitrary number of digits. Wildcards only allow us to check for a fixed number of digits or for an arbitrary number of characters (any characters).*

(d) A *palindrome* is a word or number that is the same if read forward or backward (e.g., Otto, 1991, racecar). Provide an extended regular expression that matches any palindrome with between 3 and 10 characters.

```
(.?)(.?)(.?)(.)(.)\5?\4\3\2\1
```

(e) Can the pattern in Question (d) be expressed using a basic regular expression? If so, provide such a basic regular expression. Otherwise, explain why not.

*Technically, yes, but the regular expression would have to be impractically large (albeit finite): You'd have to specify* all possible strings *that are 3–10 letter palindromes.*

*So, the answer I was looking for was that this cannot be done using basic regular expressions because they do not allow referencing back to an earlier part of the input that matches an earlier part of the pattern, which is required to match palindromes.*

**Question 5 (Automating workflows in the shell)** **9 marks**

(a) Which of the following commands can be used to determine the size of the head command? Recall that `which cmd` finds the program file that is executed when running the command `cmd`. Here, it would print `/bin/head` or `/usr/bin/head` to stdout, depending on where head is installed on the system.

```
which head | ls -l
```
☐

```
ls < which head
```
☐

```
ls -l `which head`
```
☑ ✓

```
echo `which head; ls -l`
```
☐

(b) In scripts, we may not want to know which lines in a file match a given pattern; we only want to know whether there *exists* a line in a given file that matches a given pattern. Which of the following commands prints "0" (and nothing else) if and only if `file` contains a line that matches the regular expression "a[0-9]+z"? (The `-q` option of `grep` supresses its output. `grep` has exit code 0 if there is a match, and a non-zero exit code otherwise.)

```
echo `grep 'a[0-9]+z' file`
```
☐

```
grep -q 'a[0-9]+z' file; echo $?
```
☑ ✓

```
grep 'a[0-9]+z' file; echo $STATUS
```
☐

```
fgrep -q 'a[0-9]+z' file; echo $?
```
☐

(c) Which of the following are valid bash control constructs? (There are two.)

```
for (i = 0; $i < 10; i = $i + 1); do echo $i; done
```
☐

```
for f in *.txt; do cp $f $f.bak; done
```
☑ ✓

```
if [ -f ~/.config ]; then
    prog ~/.config
else
    prog /etc/config.default
fi
```
☑ ✓

```
case var in
    1: echo "True"
    *: echo "False"
esac
```
☐

(d) Give a brief description of what the first correct control construct in Question (c) does.

*The first construct is a loop that iterates over all files with extension* `.txt` *in the current directory. For each such file* `file.txt`*, it creates a backup copy with name* `file.txt.bak`.

(e) Give a brief description of what the second correct control construct in Question (c) does.

*The second construct checks whether the file* `.config` *in my home directory exists. If so, it runs* `prog` *with this config file as argument. Otherwise, it uses the system-wide default config file stored in* `/etc/config.default`.

## Question 6 (Problem 1: Pipelines) 5 marks

You are given a database of employees of a company. The database is stored in a text file. Each line of the text file stores information about one employee. The information stored for each employee is

Field 1: Last name
Field 2: First name
Field 3: Job title
Field 4: Department
Field 5: Salary
Field 6: Year when hired

The fields on each line are separated by commas, that is, the database is a CSV file. The first three lines of such a file may look like this:

```
database.csv
Eaglewood,Bob,Technician,Tech Support,60000,2014
de Vos,Joanna,CEO,Management,20000000,1980
Lannigan,Jeffrey,Head of Sales,Sales,500000,1995
...
```

Write a one-line shell command using the commands `cut`, `sort`, and `head` that prints the last name and job title of the 5 highest-paid employees of the company.

On the file above, the output would look like this (assuming all other employees represented by ... earn less than $60,000/year).

```
de Vos,CEO
Lannigan,Head of Sales
Eaglewood,Technician
...
```

```
sort -t, -k5nr database.csv | cut -d, -f1,3 | head -5
```

## Question 7 (Problem 2: Shell scripting) 5 marks

The `diff` command compares two files and, if they are text files, displays their differences. If at least one file is a binary file, `diff` only displays the message `Files ... and ... differ`. If the two files differ, the exit code of `diff` is 1; otherwise, it is 0.

In the space below, provide a shell script `count_diffs.sh` that meets the following specifications:

- The script checks that it is given exactly two command line arguments and that each is the name of a directory. If these conditions are not met, it should print a message `USAGE: count_diffs dir1 dir2` and exit.
- Given the two directories, it inspects all *regular files* in both directories and counts the number of such files that
  - Exist only in the first directory,
  - Exist only in the second directory or
  - Exist in both directories but differ.
- The only output the script should produce is a single number: the number of differing files that were found. For example, if there is one file in the first directory that does not exist in the second directory, two files in the second directory that do not exist in the first directory, and three files that exist in both directories but whose contents differ, the output should be "6".

  Since `diff` prints at least the "Files ... and ... differ" message if two files are different, you need to find a way to silence it. Output redirection is a good way to do this.

```
dir1=$1
dir2=$2
diffcount=0
for f in $dir1/*; do
  if [[ -f $f ]]; then
    name=`basename $f`
    if [[ -f $dir2/$name ]]; then
      diff $f $dir2/$name > /dev/null
      (( diffcount = $diffcount + $? ))
    else
      (( diffcount = $diffcount + 1 ))
    fi
  fi
done
for f in $dir2/*; do
  if [[ -f $f ]]; then
    name=`basename $f`
    if [[ ! -f $dir1/$name ]; then
      (( diffcount = $diffcount + 1 ))
    fi
  fi
done
echo $diffcount
```

**Extra space for Question 7**