HW 1: Unix Basics
100 Points
2024 Spring

Name: Umberto

Questions

(9 points) What are the three main components of a Unix system, according to the notes? Give a short description of each.

User space: defines the applications, libraries and standard utilities that are user accessible.
Kernel space: this refers to the operations of OS that manage the interface between user actions and the hardware.
Hardware: the underlying physical components of a computer.

(14 points) Consider the `ls -l` output below. Label each section indicated.

```
.- This indicates that "demo" is a directory
|    .- Read/write/execute permissions          .- Directory name
| ___|___      .------------ Owner               |
v/       \     V (group)                         V
drwxr-xr-x 2 door scs  4096 2013-12-22 10:57 demo/
-rw-r--r-- 1 door scs 13454 2013-12-22 10:56 text.dat
                       ^    \_______________/    ^
   File size ----------'         |               '- File name
                                 |
                                 |
        Last modified -----------'
```

(16 points) Look up the `which` command in the manual pages (`man which` at the terminal), using either the WSL terminal or a lab machine terminal. Reading the description, what is the purpose of this command?

It searches the PATH for executable files matching the names of the arguments, returning the pathnames of the files (or links) which would be executed in the current environment, had its arguments been given as commands (it locates a command).

(10 points) Look up the `tac` command in the man pages. Describe its operation and give an example usage.

| |
|---|
| It concatenates and prints files in reverse. It writes each file to standard output, last line first. With no file, or when file is -, read standard input.<br>It might be useful to print an entire file starting from the bottom. |

(15 points) What are the three guiding principles of the Unix design philosophy, according to the notes?

| |
|---|
| Write programs that do one thing and do it well. |
| Write programs to work together. |
| Write programs to handle text streams, because that is a universal interface. |

(12 points) What are the primary purposes of *standard input*, *standard output*, and *standard error*?

| | |
|---|---|
| Standard Input (`stdin`) | Standard input is the primary input stream for reading information printed to the terminal. |
| Standard output (`stdout`) | Standard output is the primary output stream for printing information and program output to the terminal. |
| Standard error (`stderr`) | Standard error is the primary error stream for printing information to the terminal that resulted from an error in processing or should not be considered part of the program output. |

(15 points) Consider the following command line with redirects:

```
$ grep PA < sample-db.csv 2> oops > sample-db.PA.csv
```

| | |
|---|---|
| What is the *output* file? | `sample.db.PA.csv` |
| What is the *input* file? | `sample-db.csv` |
| What is the *error* file? | `oops` |

(9 points) Why is it necessary to have both *standard error* and *standard output*? Consider this example pipeline in your answer (goodfile exists but badfile does not):

```
$ cat badfile goodfile | wc
```

| |
|---|
| Because in case we are piping the output of a command to another command, we need a way to:<br>a. Still print and be aware of potential errors.<br>b. Not "pollute" the new stdin of the command with error outputs.<br>In the example case, since badfile doesn't exist, an error will be printed to the shell through stderr, while the wc command will only count words from goodfile, not including the error message. |