Homework 1

Matt Schreiber
CS311 - Operating Systems,
Winter Quarter 2014

January 13, 2014

Question 2.

Revision control systems, within the context of information science, are programs which manage changes to made to source code and associated information such as documentation. Among other things, they provide the ability to track and merge changes across one or more branches of code, they help manage source file access and editing permissions, and can assist in reverting 'deltas' (changes), restoring the source to an earlier state.

Git
Commands: `add`, `pull`, `clone`

svn (Subversion)
Commands: `diff`, `commit`, `patch`

CVS (Concurrent Versions System)
Commands:`log`, `checkout`, `release`

RCS
Commands: `ci`, `rcsdiff`, `rcsfreeze`

Question 3.

Redirection in Unix and Unix-like operating systems is the substitution of a file or stream for standard input, output, or error streams. In contrast, pipes connect the output of a program or utility to the input of another program or utility.

As an example, consider the command:

```
sed 's/\(.*\)/\u\1/' < file.txt | sort}
```

The '<' causes the contents of '`file.txt`' to be read into `sed`, which typically takes its input from `STDIN` (although the '<' is actually unnecessary since `sed` can

be invoked with a file as its argument). `sed` changes to uppercase the beginning letter of each line, printing the altered line to `STDOUT`. The pipe then connects this to `STDIN` for the `sort` utility, which sorts the lines in code-point order and prints them to `STDOUT`, which in this case means printing them to the console.

Question 4.

Here's one command which runs the `file` command on every regular file within the current file system sub-tree, taken from the `find` manpage:

```
find . -type f -exec file '{}' \;
```

Here's a way that might be faster:

```
find . -type f -print0 | parallel -0 -I arg file arg
```

Here's another way just for giggles:

```
find . -print0 | xargs -0 -I arg sh -c '[ -f arg ] && file arg'
```

Question 5.

The `make` utility builds programs and libraries from source code according to instructions specified in textfiles known as makefiles. Much of `make`'s usefulness derives from the fact that it automates parts of the build process: rather than having to manually execute a series of shell commands to build a piece of software, the programmer instead has only to specify the dependencies of the build targets, as well as the options which should be invoked with the compiler and linker, and `make` handles nearly everything else. `make` also eases the process of compiling software from multiple source files by recompiling only those targets which were affected by source code changes, rather than recompiling the whole corpus.