**Command-Line Skills - Stream Redirection and Piping**

**Activities**

Open up your command line. Each activity below consists of sets of commands to execute on the command line, interspersed with questions and commentary to discuss with your partner.

**Activity 1: Output redirection, cat, and writing files**

To begin, let’s do a refresher of some basic commands. Execute all three of the following:

cd ~

pwd

ls

Discuss with your partner: what does each of these commands do? Each name (cd, pwd, ls) stands for a longer name or phrase; what are they?

Now let’s really get into it.

ls > file\_list.txt

ls

Notice that there is now a new file in your homedir: file\_list.txt. Open this file in your text editor. What’s the text in it? How did it get there?

The > operator is the shell *output redirection operator*. It takes the output from a command, which would ordinarily be printed out on the screen, and instead writes it to a file. This stream of output is called **“standard output”**, or “standard-out” or even “stdout” for short.

If you want to see the contents of a file without leaving the command line, you can use the cat command instead:

cat file\_list.txt

When given the name of a file as an argument, cat reads the contents of that file, line by line, and prints each line to stdout.

What if we run cat with no arguments?

cat

Nothing happens! Try typing in some text, perhaps…

You do not have to be good.

You do not have to walk on your knees

for a hundred miles through the desert repenting.

You only have to let the soft animal of your body

love what it loves.

What happens each time you hit Enter? cat just parrots back to you what you typed in.

That’s because, when you don’t give cat a filename as an argument, it assumes the file will be given to it via **“standard input”**, which by default is read from the keyboard. As cat receives each line of the “file” (that is, each line of standard input as you type it), it copies it to standard-out. (The short names of standard input are “standard-in” and “stdin”.)

To finish working with cat cleanly, we have to send the “end-of-file” signal. Hit Enter so you’re on a blank line, and then hold down the Control key (*not* Command!) and type D. (This is usually abbreviated ^D in written instructions.) ^D sends the end-of-file signal, which makes cat believe it’s at the end of the file it’s copying. So, with its work concluded, it quits and now we’re back on the command line.

Since cat copies its input (from stdin) to stdout, we can use output redirection to type a file directly on the command line! Try this:

cat > poem.txt

Type in a couple lines of your favorite poem and then close out the file by sending an end-of-file signal (also known as an “EOF”) on an empty line.

ls

cat poem.txt

What happened? To recap, discuss:

* What are “standard-out” and “standard-in”?
* What does it mean to “redirect standard-out”? How do we do that on the command line?
* What is the basic functionality of the cat command? How does its behavior differ when you give it a filename as an argument and when you give it no arguments?

**Activity 2: history, grep, less, and piping**

Enter this command:

history

Discuss: what does the history command do?

What if we wanted to find all the git commands we’d entered in the past? We could write the history to a file and then use grep to search through that file for lines that contain the text “git”:

history > history.txt

grep git history.txt

Turns out, if you don’t give grep a filename, it works similarly to cat: it reads its input from standard-in instead of from a file. Try it out:

grep git

Type in some lines of text:

This line is just stuff

Here I'm talking about git though

Now I'm not

Once again, git is relevant

Pizza is so yummy

As before, to finish the virtual “file” that you’re typing into standard-in, type ^D (Control-D) on an empty line.

Now, rather than writing the output of history to a file and then using grep on that file, what if we just sent history’s stdout to grep’s stdin? We do that using the **pipe operator**:

history | grep git

What’s going on here? Discuss with your partner and try to make sense of it.

Let’s try something even more advanced:

history | grep git > git-history.txt

Use ls and cat to investigate this new git-history.txt file. What did that last command do?

When files are really big, rather than printing out their entire contents, we can use less to interactively scroll through them. Your history.txt file is probably huge. So let’s look at it with:

less history.txt

You can use the up- and down-arrow keys to scroll through the file. Type Q to quit.

less is yet *another* command that takes its input from stdin if you don’t provide a filename, so you can do:

history | less

This lets you scroll through the history without writing an intermediate file. Finally, discuss what this command does:

history | grep git | less