Special Characters:

Return, tab, and space are called blanks/whitespace

To *escape*/quote special characters use a ‘\’ (backslash) or use single quotations ‘’. For example, this \\*\\* == \*\* and ‘\*\*’ also == \*\*

Special cases: erase character (ctrl-h), line kill character (ctrl-u), etc. Must use precede with control-v to escape, \ and ‘’ doesn’t work.

Files – Ordinary and Directory:

Working directory is the directory you’re currently in, by default it is your home directory. Use pwd to Print your Working Directory.

The Command Line:

Command means both characters typed and the program that is invoked

Command line = a simple command, pipeline, or a list

Syntax = command [arg1] [arg2] … [argn] return

Not all commands require arguments, some don’t even allow arguments

Options are a special type of arguments and are preceded by -- or -, they modify the effects of a command. They are optional (hence options) and are specific to and interpreted by the command line calls not by the shell. Can have one or multiple options, usually after a command and before any other arguments. Example of an option: --help

If filename == “-l” then use “ls -- -l” to differentiate it from “ls -l”

Usage messages shows up when a command that needs arguments are used without any arguments/incorrect arguments/wrong number of arguments

An argument == token that command works on

Token == any character that’s not a blank

Command lines are processed as a whole line and then broken up accordingly to whitespaces and then processed part by part.

There are three ways to specify a file that is needed to be processed: absolute, relative and simple. The first two makes the shell look in the specified directory, the last one makes the shell look through a list of directories (PATH) for a file with the name.

Process == execution of a command by Linux, while waiting for the process to finish the shell will sleep, an inactive state. The shell only passes command line arguments to the program, it doesn’t process them, so it can’t check them for any errors. All error indicators come from the program; some utilities ignore bad options.

Standard output is a place the program can send information to, by default, it’s your screen

Standard input is the place the program gets information from, by default, it’s the keyboard

There’s also standard error, which are error messages, it’ll output to the screen.

Device file represents a peripheral device, like a printer, terminal, etc. it resides in the file structure, usually in the /dev directory.

Redirection for output is indicated with a >, a redirect output symbol. For example: cat > sample.txt, whatever is being entered at the keyboard will then be directed into the text file, instead of the screen. Redirecting output can destroy a file: cat orange pear > orange will destroy the original orange file and replace it with pear contents. To fix this, use cat to put both pear and orange into a temporary file and then use mv to rename it.

Redirection for input is indicated with a <, a redirect input symbol. Syntax is command [arguments] < filename.

Noclobber prevents overwriting a file during redirection. To use, set -o noclobber, to unuse, set +o noclobber. When set, an error message will appear when redirecting into an existing file and command won’t execute. In tsch and bash, a new file, noclobber, will be created using the touch command and the command will redirect output into that file.

Appending output to a file uses >> as the symbol. It concatenates two files. /dev/null makes the output data disappear. Pipelines, |, does redirection in a single line. A filter takes in a stream of input data and then outputs a stream of data, example, the sort command.

Foreground commands take place one after another, the shell waits for one to finish before another one stars. Background commands can continue running even when another command stars up. You can have one foreground job but many background jobs running. To begin a background command type an & (ampersand) sign before hitting return, the job will then be assigned a job number and a process identification number (PID), these numbers are the job’s identification. To suspend a foreground job, hit ctrl-z and any input from the keyboard will be suspended. Only foreground jobs can take input from the keyboard. A suspended job can be placed into the background with the command bg, to bring it back to the foreground type fg or a % followed by the number of jobs needed to be brought back into the foreground. To kill a background job, use kill followed with the PID number, not ctrl-c.

When a filename has a special character/metacharacters, they are called wildcards, since the shell can use the special characters to find random files with those characters (ambiguous file references), the process that the shell performs on those files are called globbing/pathname expansion. Some important special characters: ? (represents a single character), \* (represents zero or more characters), [] (character class, a list of characters in the brackets).

A builtin is a utility that is built into the shell, they are used the same way as utilities. When running a builtin, the shell does not fork a new process.