CSE 1320 - Intermediate Programming in C

UNIX

Dr. Alex Dillhoff

University of Texas at Arlington

UNIX Operating System

- Developed in the 1970s by a team at Bell Labs led by Ken Thompson and Dennis Ritchie
- Written mostly in C
- Originally designed as an OS for programmers
- Now exists as a multi-user, multi-tasking OS

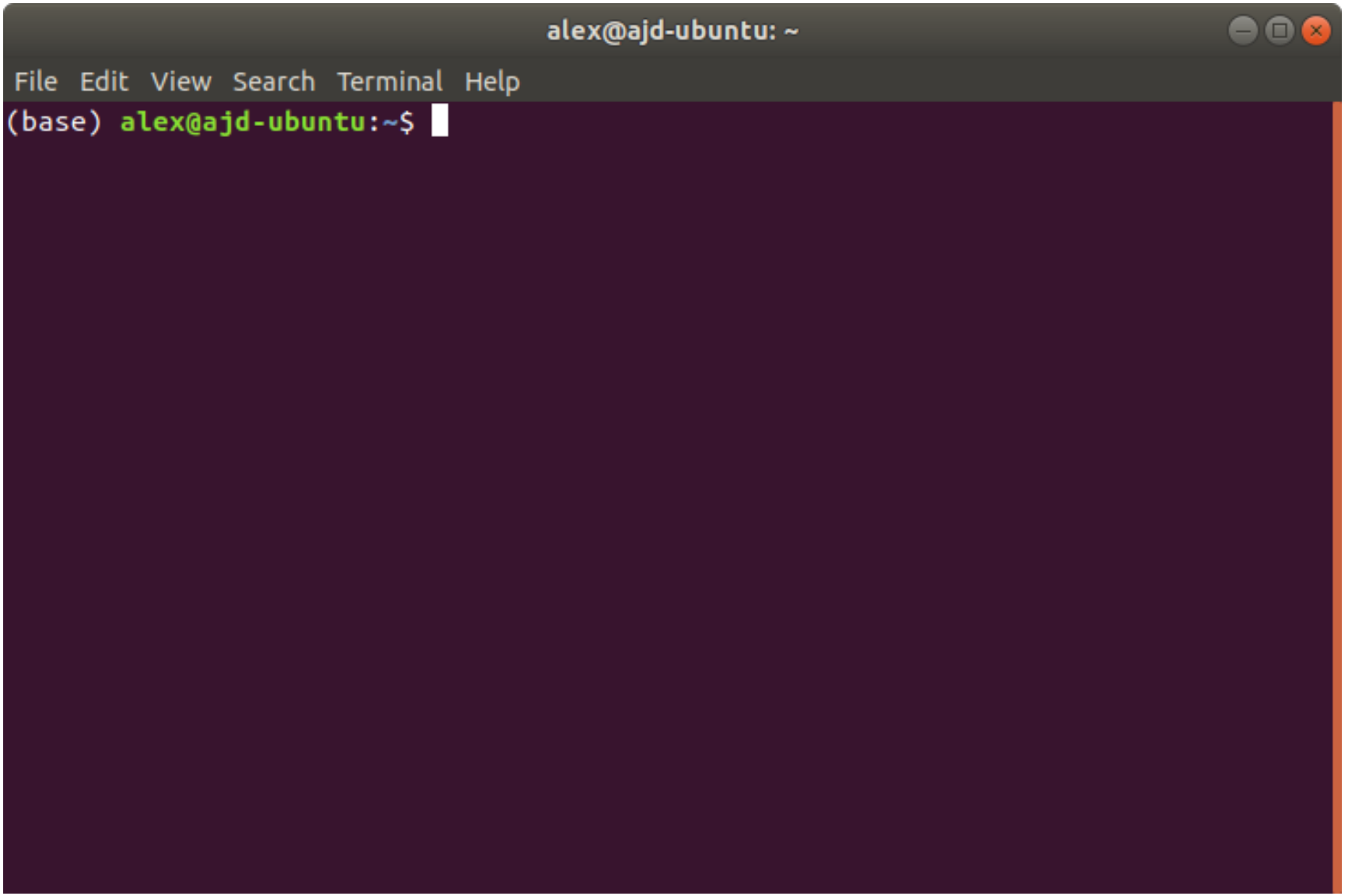
*NIX

- Many UNIX-like OSes exist today
  - Red Hat
  - CentOS
  - Ubuntu
  - Fedora
  - ...
- Exist and developed depending on specific needs, targets, etc.

Virtual Machine

If you are not already running a UNIX-like OS, please set up a Virtual Machine using the guide in Canvas (Modules/Resources).

The Shell


alex@ajd-ubuntu: ~
File Edit View Search Terminal Help
(base) alex@ajd-ubuntu:~$

The Shell

The UNIX Shell is ...

- a command line interface
- a scripting language
- a way to control the OS through scripts

UNIX Basics

- Common Commands
  - cat - concatenate files and output
  - more - file perusal 1 screen at a time
  - cp - copy files and/or directories
  - mv - move files and/or directories
  - rm - remove files and/or directories
  - mkdir - make directory
  - rmdir - remove empty directory
  - clear - clear terminal screen
  - man (tldr) - manual entry
  - cd - change current working directory
  - pwd - print working directory
  - ls - list directory contents

UNIX Basics

Where are these commands?

- `which` - locate a command

```
$ which ls
$ /bin/ls
```

grep

One of the most useful tools available in UNIX is `grep`.

It searches through files for patterns using regular expressions.

We can use it to easily filter out irrelevant information from code or log files.

grep

Filter a text file for lines that contain the word "DEBUG".

log.txt

```
DEBUG this line has it
this line doesn't
```

Terminal

```
$ grep DEBUG log.txt
DEBUG this line has it
```

grep

Regular expressions can become complicated and are outside of the scope of this course.

It is still important to be familiar with what `grep` can do.

What makes `grep` even more useful it when it is combined with pipes.

Pipes

- Inter-process communication using message passing
- Output of one process is passed as input to next process
- `proc1 | proc2`

Pipes

Example: List all processes that include `ssh`.

```
ps | grep ssh
```

- `ps` - snapshot of current processes
- `grep` - print lines matching a pattern

Streams

In Bash, there are three main streams for input and output:

- `0` - `stdin`: standard input
- `1` - `stdout`: standard output
- `2` - `stderr`: standard error

Redirection

Input and output and be redirected using `n>` and `<`

- `n>`: `n` is the file descriptor, 1 by default
- `2>&1`: redirects `stderr` to `stdout`
- `&>`: shorthand for `2>&1`

Redirection Example

Redirect output of process list to log file.

```
ps -ef > log.txt
```

End of Line Conversions

Different platforms use different ways to indicate the end of a line.

- Carriage Return (CR)
- Line Feed (LF)
- Early OSs used CR+LF
- Windows adopted CR+LF from CP/M for compatibility
- *NIX and OSX use LF
- Early Mac OS used CR

End of Line Conversions

- Carriage Return
  - **Escape Sequence:** \r
  - **Hex:** 0D
  - **Decimal:** 13
- Line Feed
  - **Escape Sequence:** \n
  - **Hex:** 0A
  - **Decimal:** 10

End of Line Conversions

End of line formats can be converted using pipes.

```
cat file.txt
    | tr '\r' '\n'
    | tr -s '\n'
    > newfile.txt
```

What is going on here?

End of Line Conversions

First, `cat file.txt` prints the contents of `file.txt` to the terminal.

This output is piped to the command

```
tr '\r' '\n'
```

translating all occurrences of `\r` to `\n`

End of Line Conversions

This translated output is then piped into

```
tr -s '\n'
```

removing duplicate, consecutive occurrences of `\n`.

End of Line Conversions

Lastly, the command

```
> newfile.txt
```

redirects the output to the file `newfile.txt`.

End of Line Conversions

This is certainly not the only way to perform EOL conversions.

It is a common enough action that someone wrote a tool to do it with one command.

Simpler way: `dos2unix, unix2dos`

What next?

Mastering UNIX is well beyond the scope of this course.

Through regular usage, you will start to pick up on more useful commands and their applications.

A decent overview of basic commands can be found here: `http://mally.stanford.edu/~sr/computing/basic-unix.html`