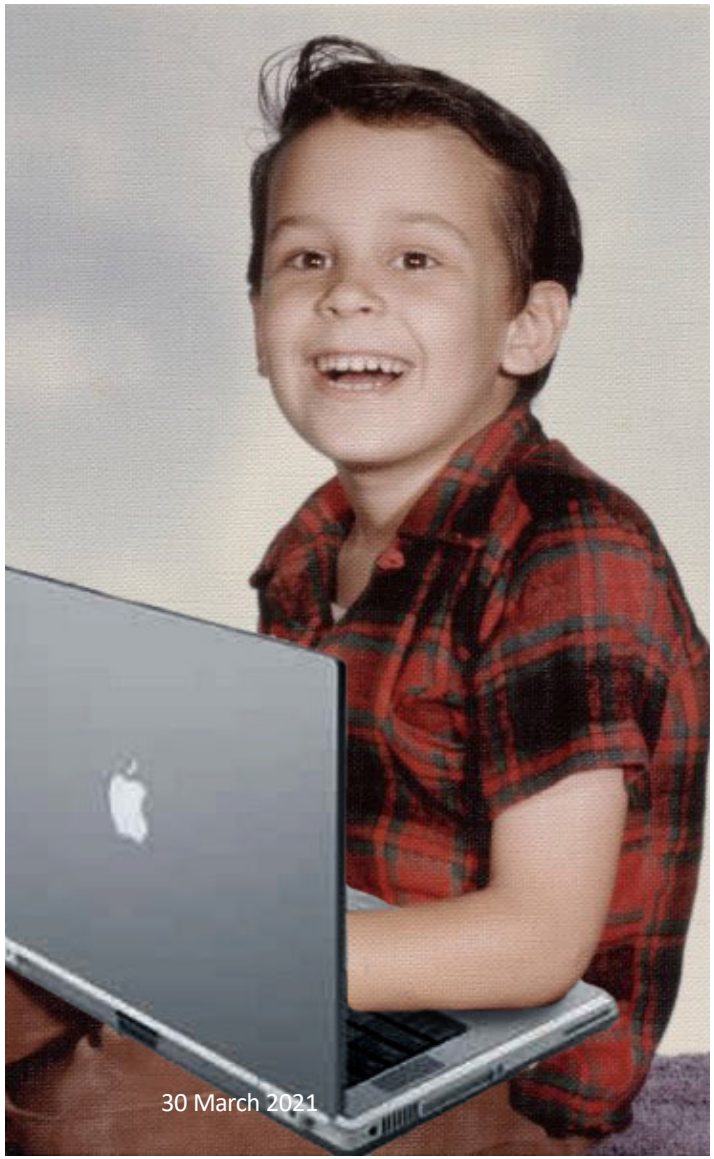
30 March 2021

Getting Started

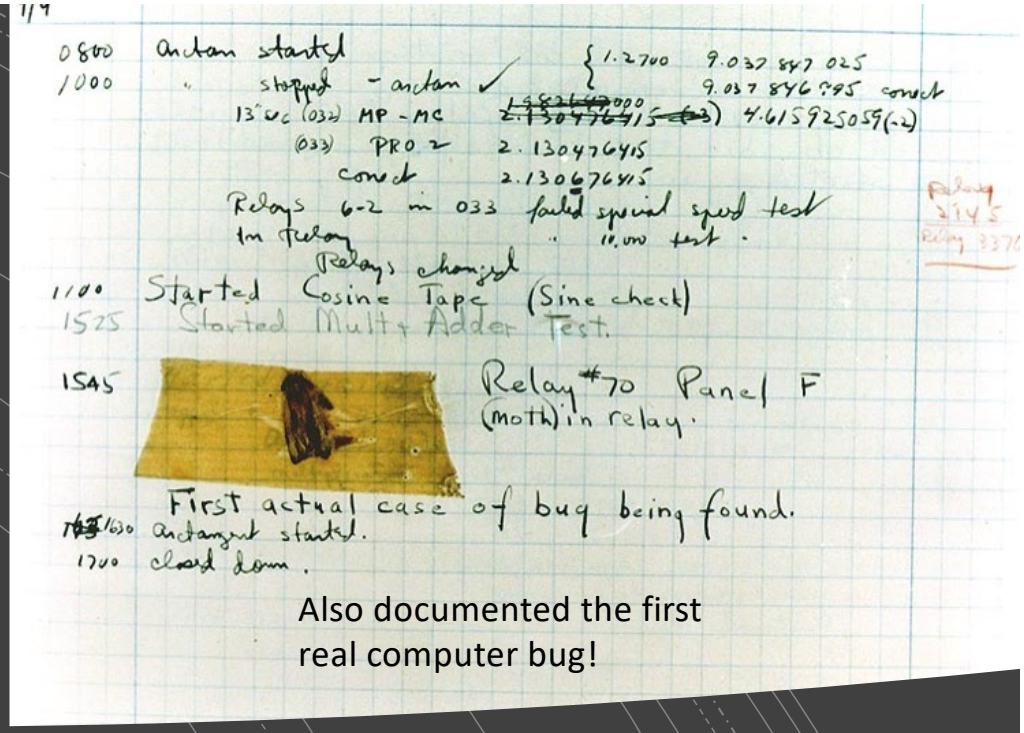
Prof. Darrell Long

CSE 13S



Assumptions

- You have access to a computer.
- You have a virtual machine running Ubuntu 20.04
- You have created your account on `git.ucsc.edu`
- You have created a `ssh` key
- You have written at least a simple in a higher-level language program.
- You have passed CSE 12.



Also documented the first
real computer bug!

Today's Tools

- Editor
- Compiler
- `git`
- Command line

Let’s get you on the virtual machine

- We’ll make an `ssh` key
- Make directories
- Clone a repository
- … And pretty much complete assignment 0

Create an `ssh` Key

darrell — -bash — 80×24

```
pascal:~ darrell$ ssh-keygen -b 384 -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/Users/darrell/.ssh/id_ecdsa):
Created directory '/Users/darrell/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/darrell/.ssh/id_ecdsa.
Your public key has been saved in /Users/darrell/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:4DQBn2kcGI44+sezmp7CIZKtAiKaV88D4rULUYq+hdE darrell@pascal.lan
The key's randomart image is:
+---[ECDSA 384]---+
|     o+o         |
| . o.o =         |
|o . o X          |
|.o.o + o         |
|o+oE  . S        |
|Oo=o+            |
|O*+=+=           |
|*.==.o+          |
|.==.o. .         |
+----[SHA256]-----+
pascal:~ darrell$
```

mkdir cse13s

darrell — ssh unix.ucsc.edu — 80×20

```
unix3.lt.ucsc.edu [106]% mkdir cse13s
unix3.lt.ucsc.edu [107]% chmod 700 cse13s
unix3.lt.ucsc.edu [108]% cd cse13s
unix3.lt.ucsc.edu [109]% ls
unix3.lt.ucsc.edu [110]%
```

Create the directory

Keep it private

Go to the directory

Anything in it?

Clone the Respository

darrell — ssh unix.ucsc.edu — 80×20

```
unix3.lt.ucsc.edu [111]% git clone git@gitlab.soe.ucsc.edu:cse013s/spring19/darrell.git
Cloning into 'darrell'...
remote: Enumerating objects: 166, done.
remote: Counting objects: 100% (166/166), done.
remote: Compressing objects: 100% (104/104), done.
remote: Total 166 (delta 66), reused 119 (delta 51)
Receiving objects: 100% (166/166), 22.46 KiB | 0 bytes/s, done.
Resolving deltas: 100% (66/66), done.
unix3.lt.ucsc.edu [112]%
```

This is your CruzID

Use the right term

Copy this URL

Where did you get that?

git.ucsc.edu

GitLab system status is available here and here

GitLab Projects Groups More Search or jump to...

darrell

Project overview
Details
Activity
Releases
Repository
Issues 0
Merge Requests 0
CI / CD
Operations
Packages & Registries
Analytics
Wiki
Snippets
Members
Settings
Collapse sidebar

CSE 13S › Spring 2021 › darrell › Details

darrell
Project ID: 4200

Star 0 Fork 0

4 Commits 1 Branch 0 Tags 164 KB Files 166 KB Storage

master darrell / + History Find file Web IDE Clone

hello.c added
Darrell Long authored 22 hours ago

Clone with SSH
git@git.ucsc.edu:cse13s/spri

Clone with HTTPS
https://git.ucsc.edu/cse13s/

README CI/CD configuration Add LICENSE Add CHANGELOG
Add Kubernetes cluster

Name	Last commit	Last update
asgn0	hello.c added	22 hours ago
asgn1	init template repo files	1 week ago
asgn2	init template repo files	1 week ago
asgn3	init template repo files	1 week ago
asgn4	init template repo files	1 week ago
asgn5	init template repo files	1 week ago
asgn6	init template repo files	1 week ago

Start Work

darrell — ssh unix.ucsc.edu — 80×20

```
unix3.lt.ucsc.edu [118]% ls
darrell/
unix3.lt.ucsc.edu [119]% cd darrell/asgn0
unix3.lt.ucsc.edu [120]% vi Makefile
```

Did it work?

Enter the directory

Edit the file

Vi/Vim

The pursuit of knowledge is never-ending. The day you stop seeking knowledge is the day you stop growing
—Brandon Travis Ciaccio

- Vi is the standard text editor found on Unix systems.
- Vim is an acronym for “Vi IMproved” and is considered a clone of the Vi editor.
  - Most UNIX systems simply alias “vi” to “vim” so using either one is fine.
- Functionally, Vim is almost a proper superset of Vi, thus everything in Vi as also available in Vim.
  - In general, use Vim when possible as includes built-in support for numerous programming languages and has been ported to a much wider range of OS’s than Vi.
- Vim uses two different modes for text editing:
  1. Normal mode (also called command mode)
  2. Insert mode
- We will only cover the basics of Vi/Vim to get you started.
  - If you wish to learn all the nuances of Vi/Vim, you will have to do so yourself.

Normal Mode

- The default mode when you open a file in Vim is normal, or command mode.
- Commands enable you to move anywhere in the file, perform edits, enter insert mode to add new text, save the file, or exit the file.
- Basic cursor movement in normal mode:
  - `h` – move cursor left one space.
  - `j` – move cursor down one line.
  - `k` – move cursor up one line.
  - `l` – move cursor right one space.
- File related commands:
  - `:wq` – save and quit file.
  - `:q!` – quit file without saving.
  - `:%s/hello/goodbye/g` – replace all instances of “hello” with “goodbye”.
  - `:`*n* – jump to line number *n*.

Commands

- Vim commands can be thought of as a language it and of itself.
- For example, in Vim, ‘`d`’ means to delete, and ‘`w`’ means a word:
  - These operations can be combined into the command “`diw`”.
  - This reads as “delete in word”, and deletes the word your cursor is in.
- You can also combine operations with movement/motion commands (`h`, `j`, `k`, `l`) and counts:
  - The command “`dl`” deletes one character to the right.
  - The command “`3dl`” deletes three characters to the right.
- This ability to combine operations with motions and counts is what makes Vim so flexible and great for text editing.
- Note that Vim is case-sensitive, so make sure you’re using the right casing for your commands.

Insert Mode

- Insert mode allows you to add text to a file.
  - Normal mode is used to navigate and make quick edits to files.
  - Insert mode is used only to add text.
  - You should spend most of your time in normal mode.
- Insert mode is entered when the key '`i`' is pressed in normal mode.
  - You can also enter insert mode with '`a`', which appends text after the cursor rather than before.
- Insert mode is exited when the escape key is pressed.

Basic Vim Cheat sheet

- `H` – move to top of screen
- `M` – move to middle of screen
- `L` – move to bottom of screen.
- `w` – jump forwards to start of word
- `e` – jump forwards to start of word
- `b` – jump backwards to start of word
- `%` – move to matching character (used for braces)
- `^` – jump to first non-blank character in line
- `gg` – go to first line of document
- `G` – go to last line of document
- `:wq` – save and quit
- `r` – replace a single character
- `cc` – change (replace) entire line
- `C` – change to end of the line
- `ciw` – change in word
- `u` – undo
- `CTRL+r` – redo
- `.` – repeat last command
- `yy` – yank (copy) line into buffer
- `Y` – yank to end of the line
- `p` – paste from buffer
- `o` – open new line below current line

Editors live in your fingers…

- The commands are not obvious.
  - This is not unique to Vim, it is also true for emacs.
- It requires practice!
  - It is a kind of muscle memory, like playing the guitar.
  - No one thinks “I need to type i now to insert”

vi Makefile

darrell — vi Makefile — 66×20

```
CFLAGS=-O -Wall -Werror -Wextra -Wpedantic -Wshadow
CC=clang

hello   :       hello.c
        $(CC) $(CFLAGS) -o hello hello.c

format  :
        clang-format -i *.[ch]

clean   :
        rm -f hello *.o
~
~
~
~
~
~
~
~
```

vi hello.c

tmp — vi hello.c — 66×20

```
#include <stdio.h>

int main(void) {
    printf("Hello world!\n");
    return 0;
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

vi README.md

tmp — vi README.md — 70×20

```
# Purpose

Simple to print the message "Hello world!"

# Building

Typing "make" will build it

Typling "format" will reformat the source

Typing "clean" will remove the binaries
~
~
~
~
~
~
~
~
```

```
darrell@riemann /tmp % vi Makefile
darrell@riemann /tmp % vi hello.c
darrell@riemann /tmp % make
clang -O -Wextra -Wpedantic -Wshadow -Wall -Werror -o hello h
darrell@riemann /tmp % ./hello
Hello world!
darrell@riemann /tmp %
```

Does it work?

Documentation

And further, by these, my son, be admonished: of making many books there is no end; and much study is a weariness of the flesh.
—Ecclesiastes 12:12

- There are two more required files:
  - `DESIGN.pdf`
  - `WRITEUP.pdf`
- In this case they will both be very short, but in the future, they will be more detailed.
- `DESIGN.pdf`
  - This document describes the design of your program and answers to the pre-lab questions.
  - It describes the algorithms and design decisions you have made.
  - It describes the problem that you are solving, inputs, and expected outputs.
- `WRITEUP.pdf`
  - This document contains your analysis of running your program.
  - For example, if your assignment was to compare sorting algorithms this file would contain the results of that comparison.

Submit using `git`

darrell — ssh unix.ucsc.edu — 80×20

```
unix3.lt.ucsc.edu [127]% make clean
rm -rf hello *.o infer-out
unix3.lt.ucsc.edu [128]% git add hello.c WRITEUP.pdf DESIGN.pdf README.md Makefile
unix3.lt.ucsc.edu [129]% git commit -am "First version"
[master 80d848e] First version
 2 files changed, 7 insertions(+), 3 deletions(-)
unix3.lt.ucsc.edu [130]% git push
Counting objects: 9, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 570 bytes | 0 bytes/s, done.
Total 5 (delta 2), reused 0 (delta 0)
To git@gitlab.soe.ucsc.edu:cse013s/spring19/darrell.git
   847b5b1..80d848e  master -> master
unix3.lt.ucsc.edu [131]%
```

Add the files

Commit your changes

Push to the server

```
darrell — ssh unix.ucsc.edu — 80×20
unix3.lt.ucsc.edu [133]% git add CHEATING.pdf
unix3.lt.ucsc.edu [134]% git commit -am "I have read the cheating policy"
[master 6042a88] I have read the cheating policy
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 asgn0/CHEATING.pdf
unix3.lt.ucsc.edu [135]% git push
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 318 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
To git@gitlab.soe.ucsc.edu:cse013s/spring19/darrell.git
   80d848e..6042a88  master -> master
unix3.lt.ucsc.edu [136]%
```

One last thing…

- You must read the cheating policy.
- You acknowledge having done so by submitting a PDF of the policy using `git`.
- If you do not read and acknowledge the policy then your assignments *will not be graded*.
- Do this *once* for `assignment 0`.