

# CS 241

## Data Organization using C

### *Hello World – in Linux*

Instructor: **Joel Castellanos**

e-mail: [joel@unm.edu](mailto:joel@unm.edu)

Web: <http://cs.unm.edu/~joel/>

Office: Farris Engineering Center  
(FEC bldg. #119)  
Room 2110

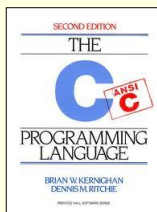


```
% gcc helloWorld.c
```

8/22/2019

1

## Read: Kernighan & Ritchie



- Due Thursday, Aug 22
  - 1.1: Getting Started
  - 1.2: Variables and Arithmetic Expressions
  - 1.3: The For Statement
  - 1.4: Symbolic Constants

2

2

## CS Building & Website: <https://www.cs.unm.edu/>

The University of New Mexico UNM A-Z

**1**

**2**

**3**

**Farris Engineering Center**

- CS Faculty offices
- CS Department office
- CS Help and Support
- CS Labs
- CS Tutors

3

3

## CS Computer Account <https://www.cs.unm.edu/>

The University of New Mexico UNM A-Z StudentInfo Faculty myUNM Directory Search

Support | News | Find People | Contact Us | Location

Search CS Search

**COMPUTER SCIENCE**

ABOUT US ACCREDITATION PROGRAMS & DEGREES RESEARCH STUDENTS

UNM > Home > Computer Facilities > Contact Support

**Computer Facilities**

DESCRIPTION OF FACILITIES

FACILITIES

POSTERS

CHANGING YOUR PASSWORD

**CONTACT SUPPORT**

FAQ

EMAIL

**Contact Support**

The Computer Science System Support Group provides IT help for the department faculty and staff. We use the Request Tracker ticketing system to provide support through email, please send email to [cssupport@cs.unm.edu](mailto:cssupport@cs.unm.edu). This is the preferred way to get help as either Rudy or Luke will get the email and be able to provide better support than email us directly.

The SSG provides walk-in assistance in FEC 3550.

**Hours**

The office is usually staffed during normal business hours (9 am to 4 pm). In general, if the light is on, step into our offices – someone will likely be able to assist you.

**Phone Numbers and Contact Addresses**

Walk-in assistance in FEC 3550

4

4

# CS Computer Account <https://www.cs.unm.edu/>

The screenshot shows a web page with a navigation bar at the top containing links for ABOUT US, ACCREDITATION, PROGRAMS & DEGREES, RESEARCH, and STUDENTS. On the left side, there is a sidebar menu with categories like Computer Facilities, DESCRIPTION OF FACILITIES, FACILITIES, POSTERS, CHANGING YOUR PASSWORD, CONTACT SUPPORT, FAQ, EMAIL, GETTING STARTED (highlighted), INSTALLED PROGRAMS, MAKING YOUR HOME PAGE, and POLICIES. The main content area is titled 'Getting Started' and includes sections for 'After getting an CS account', 'Working Remotely', and 'Wireless networks'. The 'After getting an CS account' section explains that users will receive a new email and storage account, and provides details about the computer lab location (room 309) and equipment (HP LaserJet, queue name fec389, and a color LaserJet at fec389c). The 'Working Remotely' section lists three groups of computers (shuttle.cs.unm.edu, moons.cs.unm.edu, glgs.cs.unm.edu) and provides instructions for connecting via SSH or PuTTY. The 'Wireless networks' section is partially visible at the bottom.

5

5

## Hello World: A Program in C

```
moons.cs.unm.edu - PuTTY
1 #include <stdio.h>
2 int main(void)
3 {
4     printf("Hello World\n");
5     return 0;
6 }
~
~
~
-- INSERT --
```

{ } Curly Brackets are used to group lines of code into blocks. Lines 3 through 6 make the body of "main".

( ) Parenthesis are used for function parameters. In line 2, the parameter list for "main" is "void".

6

6

## Step 1A: Secure Shell to \*.cs.unm.edu

- **PuTTY**: free implementation of Telnet and SSH (Secure Shell network protocol) for Windows 95, 98, ME, NT, 2000, XP and Vista on Intel x86 platforms.
- Mac OS X comes with its own implementation of OpenSSH, so you don't need to install third-party software. From Macintosh HD and go to the *Applications* folder, then *Utilities* from within that, select *Terminal*.

- `moons.cs.unm.edu`
- `shuttle.cs.unm.edu`



7

7

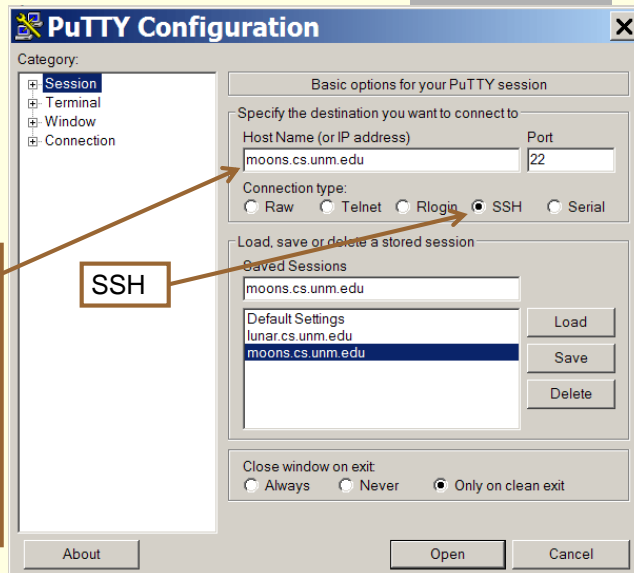
## Step 1B: PuTTY: Set Host & Protocol

- Open Putty



Host:  
`moons.cs.unm.edu`  
or  
`trucks.cs.unm.edu`

Sometimes, one of the servers is down. If one of these does not work, then try the other.



8

8

## Step 1C: PuTTY: Set Window Options

The screenshot shows the 'PuTTY Reconfiguration' dialog box with the 'Window' category selected. The 'Options controlling PuTTY's window' section is visible. The 'Set the size of the window' section has 'Columns' set to 80 and 'Rows' set to 24. The 'When window is resized' section has 'Change the number of rows and columns' selected. The 'Control the scrollbar in the window' section has 'Lines of scrollbar' set to 200. Callout boxes point to the 'Rows' field, the 'When window is resized' section, and the 'Lines of scrollbar' field.

May want to increase **Rows** from the default 24.

When Window is resized: Change the number of rows and columns.

Lines of scrollbar

9

9

## Step 1D: PuTTY: Set Translation

Window → Translation → Received data assumed to be in which character set → **UTF-8**

The Linux Host sends a binary single to PuTTY.

PuTTY needs to be told which standard to use to translate that signal.

Most standards use the same codes for the upper and lower case 26 letters of the 26 English alphabet.

“Special” characters,  $\pi$ , ©, ±, ñ Often have different codes in different standards.

The screenshot shows the 'PuTTY Configuration' dialog box with the 'Translation' category selected. The 'Options controlling character set translation' section is visible. The 'Received data assumed to be in which character set' dropdown is set to 'UTF-8'. The 'Adjust how PuTTY handles line drawing characters' section has 'Use Unicode line drawing code points' selected. A callout box points to the 'UTF-8' dropdown.

**Good:** hello.c:24: error: 'i' undeclared

**Bad:** hello.c:24: error: âiâ undeclared

10

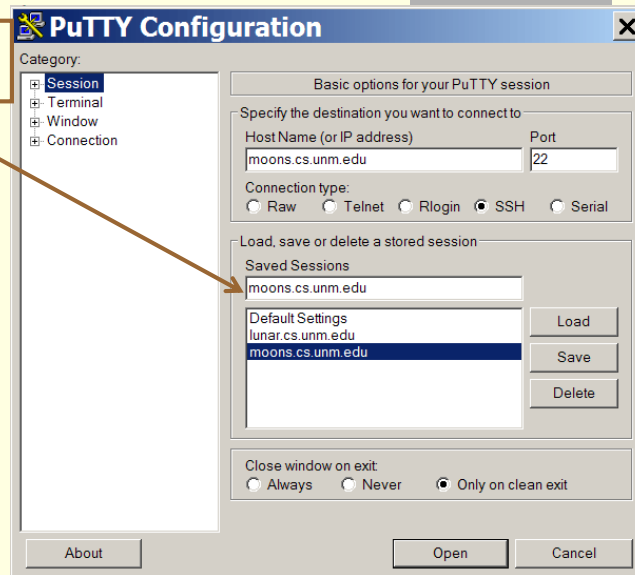
10

## Step 1E: PuTTY: Save Configuration

Give your configuration a name and click **Save**.

You can also customize other parts of the configuration:

- Default User Name,
- Font Size,
- Foreground Color,
- Background Color,
- etc.

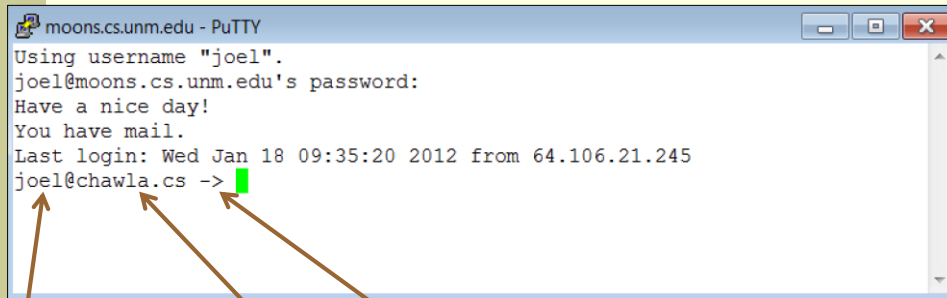


11

11

## Step 1F: Open the Connection

**bash Shell:** Default Linux shell environment on cs machines



User name

Command Prompt

Machine Name

12

12

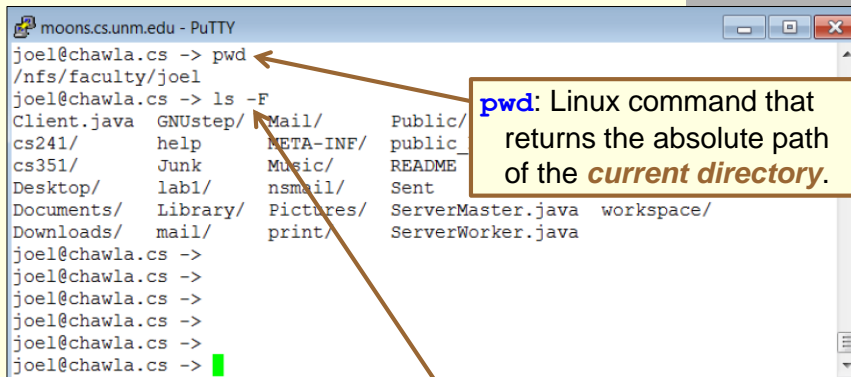
## Aside: Unix Shells

- A **Unix shell** is a command-line interpreter that provides a traditional user interface for the Unix operating system and for Unix-like systems (i.e. Linux).
- The most influential Unix shells:
  - **C shell** (syntax modeled after the C programming language)
  - **Bourne shell** early de facto standard
  - **bash** (Bourne-Again SHell): Superset of Bourne Shell functionality. Default interactive shell for users on most GNU/Linux and Mac OS X systems.

13

13

## Aside: Poking Around Your Home Directory



```
moons.cs.unm.edu - PuTTY
joel@chawla.cs -> pwd
/nfs/faculty/joel
joel@chawla.cs -> ls -F
Client.java  GNUstep/  Mail/      Public/
cs241/       help      META-INF/  public_
cs351/       Junk      Music/     README_
Desktop/     lab1/     nsmail/    Sent
Documents/   Library/  Pictures/   ServerMaster.java  workspace/
Downloads/   mail/     print/     ServerWorker.java
```

**pwd**: Linux command that returns the absolute path of the **current directory**.

**ls**: Linux command that lists all files in the current directory.

**ls -F**: The '-F' is a **command line option** that tells **ls** to append a '/' character to the end of directories, a '\*' character to the end of executable files and an '@' character to the end of symbolic links.

14

14

## Step 2: Create a Working Directory



```
moons.cs.unm.edu - PuTTY
joel@oberon.cs -> mkdir lab1
joel@oberon.cs -> cd lab1
joel@oberon.cs -> pwd
/nfs/faculty/joel/lab1
joel@oberon.cs -> ls
```

**mkdir name:** Linux command that creates a directory with the given name.  
Returns an error message if a file already exists with that name.

**cd name:** Linux command that looks in the *current directory* for a directory of the given name. If the given directory is found, then the current directory is changed to the given directory.

If the **cd** command is used without specifying a name, then the current directory is changed to the user's *home directory*.

15

15

## Step 3: Open a Text Editor: vim



```
moons.cs.unm.edu - PuTTY
joel@oberon.cs -> vim hello.c
```

**vim:** text editor.

**hello.c:** File name passed to vim.  
If this file does not exist in the current directory, then a new empty file is created with the given name.

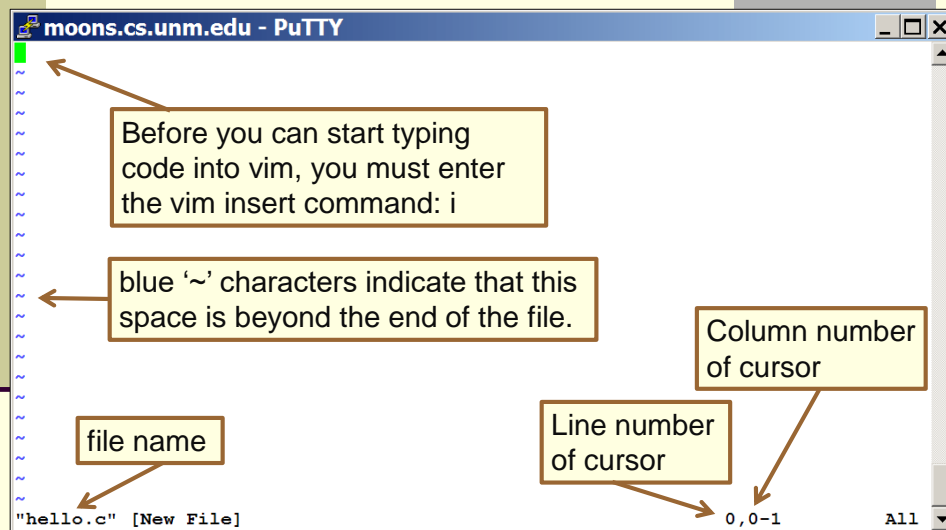
C source code should be given a file name that ends with **.c**.

16

16



## Aside: vim and an Empty File



17

17

## Aside: vim Text Editor



- vim is modal — a design choice that tends to confuse new users unaware of insert-mode.
- Run in remote graphics window.
- Run in remote text-only window.
- Free, and open source – compiled executables for **\*\*\*every\*\*\*** platform:
  - Atari 800
  - Small hardware devices.
- Very small and simple executable.
- Tons of documentation
- Official Website: <http://www.vim.org/>

*Tip: if network is problematic, download vim and run locally.*

*Then SFTP.*

18

18

## Aside: vim: 1 of 3



<b>i</b>	Enter insert mode
<b>v</b>	Enter visual select mode (Use normal movement keys)
<b>&lt;Esc&gt;</b>	Exit current mode (insert or select)
<b>y</b>	Yank selected text
<b>d</b>	Delete selected text (when in visual select mode)
<b>dd</b>	Delete line
<b>x</b>	Delete character
<b>rc</b>	Replace one character with <i>c</i>
<b>p</b> ( <b>P</b> )	Put yanked or deleted text <i>after</i> ( <i>before</i> ) the cursor.
<b>:w</b>	Write file
<b>:q</b>	Quit
<b>:q!</b>	Quit without saving

19

19

## Aside: vim: 2 of 3



<b>←↑→↓</b>	Moves cursor left, up, right or down.
<b>h j l k</b>	Moves cursor left, up, right or down.
<b>\$</b> or <b>&lt;end&gt;</b>	Moves to next '\n' character – Very useful in long (multi-line) lines: <i>In VIM, you cannot just click the mouse in the text to move the cursor.</i>
<b>0</b>	Moves cursor to beginning of line.
<b>u</b>	Undo last action. Can be stepped backward.
<b>U</b>	Undo all changes to line
<b>J</b>	Join next line to end of current line.
<b>a</b>	Append (enter insert mode, <i>after</i> the cursor).

20

20

## Aside: vim: 3 of 3



- `:=` Display current line number.
- `:n` Jump to line number  $n$ .
- `/x` Search for  $x$ , where  $x$  is a character string.
- `/` Repeat last search.
  
- `:set nu` Display line numbers.
- `:set nonu` Turn off display line numbers.
  
- `%` Jump to a matching opening or closing parenthesis, square bracket or a curly brace: ( [ { } ] )  
Jump to start or end of a C-style comment: /\* \*/.

21

21

## Step 4: Enter C Source Code



`i` (turn on INSERT mode)  
`<enter code>`

```
moons.cs.unm.edu - PuTTY
#include <stdio.h>
int main()
{ printf("Hello World\n");
  return 0;
}
~
~
~
~
-- INSERT --
```

22

22

## Steps 5, 6 & 7: Exit vim, Compile & Run

Within vim:

- i** (turn INSERT mode)
- <enter code>*
- esc** (exit INSERT mode)
- :w** (save file)
- :q** (quit vim and return to Linux command line)

```
moons.cs.unm.edu - PuTTY
joel@oberon.cs -> vim hello.c
joel@oberon.cs -> gcc hello.c
joel@oberon.cs -> ./a.out
Hello World
joel@oberon.cs ->
```

**gcc**: Runs a C compiler program with the source file hello.c as input.

If **gcc** compiles hello.c without errors, then **gcc** will produce a.out: an executable file containing machine code.

**./a.out**: runs the program.  
**./** tells Linux to look in the current directory for **a.out**.

23

23

## Syntax Error

```
moons.cs.unm.edu - PuTTY
1 int main()
2 { printf ("Hello World\n");
3 return 0;
4 }
5
~
~
~
~
:set nu
```

```
oberon 72 % gcc hello.c
hello.c: In function 'main':
hello.c:2: warning: incompatible implicit
declaration of built-in function 'printf'
oberon 73 %
```

24

24

## A Look at What We Created



```
moons.cs.unm.edu - PuTTY
setebos 148 % pwd
/nfs/faculty/joel/lab1
setebos 149 %
setebos 149 %
setebos 149 %
setebos 149 % ls -F -l
total 16
-rwxr-xr-x 1 joel ssg 10931 2010-01-27 09:46 a.out*
-rw-r--r-- 1 joel ssg 71 2010-01-27 09:45 hello.c
setebos 150 %
```

Absolute path of the current directory.

The `-l` option of `ls` specifies listing files in the *long listing format*.

owner

group

File size in bytes.

The executable is large because we did not compile with a *dynamically linked library*. Thus, `a.out` contains all code for `printf`.

Access permissions: *details later*

25

25

## View File Contents: *more*



`more filename`

DESCRIPTION

`more` is a filter for paging through text one screenful at a time. This version is especially primitive. Users should realize that `less` provides more emulation and extensive enhancements.

```
moons.cs.unm.edu - PuTTY
io 54 % a.out > longOutput
io 55 % more longOutput
```

Sometimes your program's output is too long to fit on the screen.

This is especially true when debugging.

Redirect your output to a file, `>`, then use `more` (or `less`).

26

26

## Hidden Files: `ls -a`



The `-a` option of `ls` specifies to include *hidden files*.

File names that start with `.` (period) are hidden files in Linux:

- `.` is the current directory.
- `..` is the parent directory.

`cd ..` will change the current directory to the parent directory.

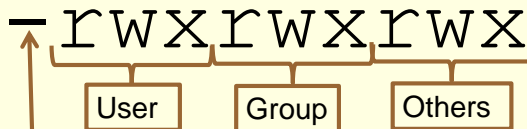
`-F` displays `/` after directories and `*` after executables.

```
moons.cs.unm.edu - PuTTY
setebos 158 % ls -F -l -a
total 28
drwxr-xr-x  2 joel ssg  4096 2010-01-27 09:46 ./
drwxrwxrwx 62 joel ssg  8192 2010-01-27 09:46 ../
-rwxr-xr--  1 joel ssg 10931 2010-01-27 09:46 a.out*
-rw-r--r--  1 joel ssg   71 2010-01-27 09:45 hello.c
setebos 159 %
```

27

27

## Linux Access Permissions: `chmod`



`d` if this is a directory file.

Only executable files should have execute permission:

- machine code,
- shell scripts,
- directories

In Linux, a file has nine independent permission properties:

*read*, *write*, and *execute*

for each of three types of users:

*user*, *group*, *others*.

`chmod <who><+|-><permission> file`

`chmod o-x a.out` //remove execute permission from others.

`chmod a-w a.out` //remove write permission from *all* users.

`chmod a+r a.out` //add read permission to *all* users.

28

28

## bash Environment Variable: \$PATH



```
moons.cs.unm.edu - PuTTY
joel@chawla.cs -> which vim
/usr/bin/vim
joel@chawla.cs ->
joel@chawla.cs -> █
```

**which**: Linux command that searches your shell's `$path` variable for the given argument.

- Returns the absolute path or "Command not Found".

```
moons.cs.unm.edu - PuTTY
joel@chawla.cs -> echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/X11R6/bin:
joel@chawla.cs ->
joel@chawla.cs ->
joel@chawla.cs ->
joel@chawla.cs ->
joel@chawla.cs ->
joel@chawla.cs ->
joel@chawla.cs ->
joel@chawla.cs -> █
```

directories delimited by :

By default, `.` (the current directory) is not included in `$PATH`.

29

29

## `.bash_profile` Configuration File



- When you login to a bash shell, either sitting at the machine or remotely via ssh, bash looks in your **home directory** for the hidden file `.bash_profile`. If this file exists, then bash will automatically execute it to configure your shell before showing you the initial command prompt.
- You can create and edit `.bash_profile`, *but be careful*.
- For example, `vim .bash_profile`

```
export PATH=$PATH:.  
alias ls="ls -F"
```

Note: the current directory at the **start** of `$PATH` is considered a security risk. Why?

The first line adds the current directory (`.`) to the **end** of the path.

The second line creates an alias so that whenever you enter `ls` as a shell command, bash will replace it with `ls -F`.

30

30

## Linux Change Directory: `cd`



`cd name` Looks in the **current directory** for a directory of the given name. If the given directory is found, then the current directory is changed the given directory.

`cd` Sets the current directory to the user's **home directory**.

`cd .` Does nothing (why?)

`cd ..` Sets the current directory to the parent directory.

`cd ~user` Sets the current directory to the home directory of the specified user.

```
moons.cs.unm.edu - PuTTY
setebos 198 % cd ~joel
setebos 199 % ls -d -l cs241 public_html
drwx----- 4 joel ssg 4096 2009-05-14 14:32 cs241/
drwxr-xr-x 22 joel ssg 4096 2010-01-15 10:47 public_html/
setebos 200 %
```

31

31

## Manual Pages: `man command`



`man command` displays manual pages on the specified command.

Usually the man pages do not fit inside the window.

`<spacebar>` moves down one page.

`<enter>` or `<down arrow>` moves down one line.

`<up arrow>` moves up one line.

`q` quit man pages.

```
moons.cs.unm.edu - PuTTY
NAME
    lpr - print files

SYNOPSIS
    lpr [ -E ] [ -H server[:port] ] [ -U username ] [ -P destina
tion[/instance] ] [ -# num-copies [ -h ] [ -l ] [ -m ] [ -o
option[=value] ] [ -p ] [ -q ] [ -r ] [ -C/J/T title ] [

Manual page lpr(1) line 5
```

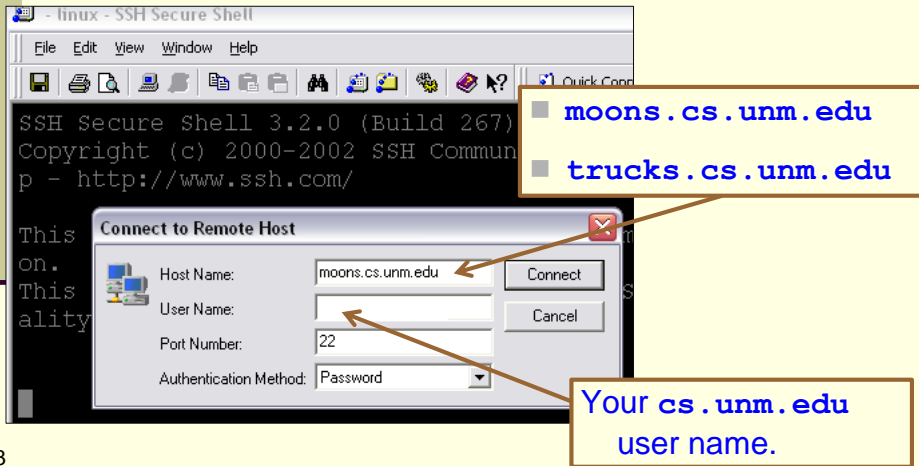
32

32



## UNM Labs: Secure Telnet Connect: 1 of 2

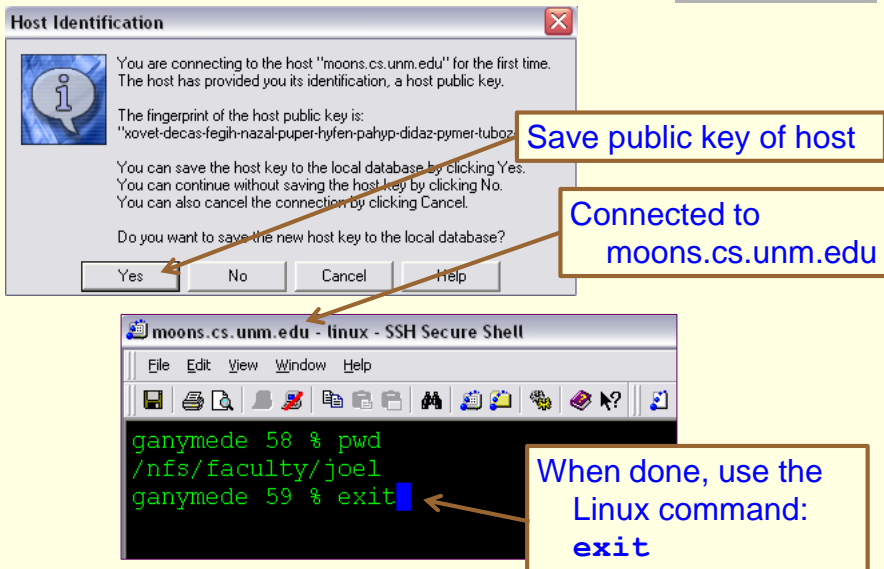
Start → All Programs → Secure Telnet and FTP  
→ Telnet



33

33

## UNM Labs: Secure Telnet Connect: 2 of 2

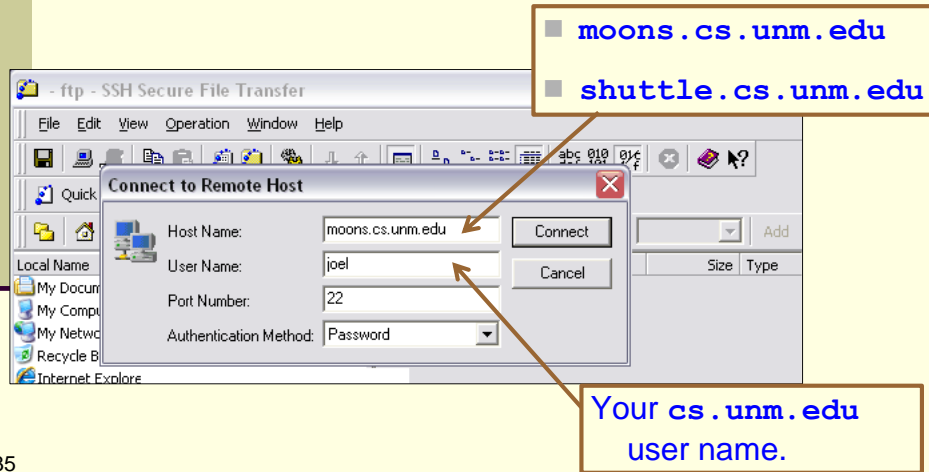


34

34

## UNM Labs: Moving files: SFTP: 1 of 2

Start → All Programs → Secure Telnet and FTP  
→ FTP



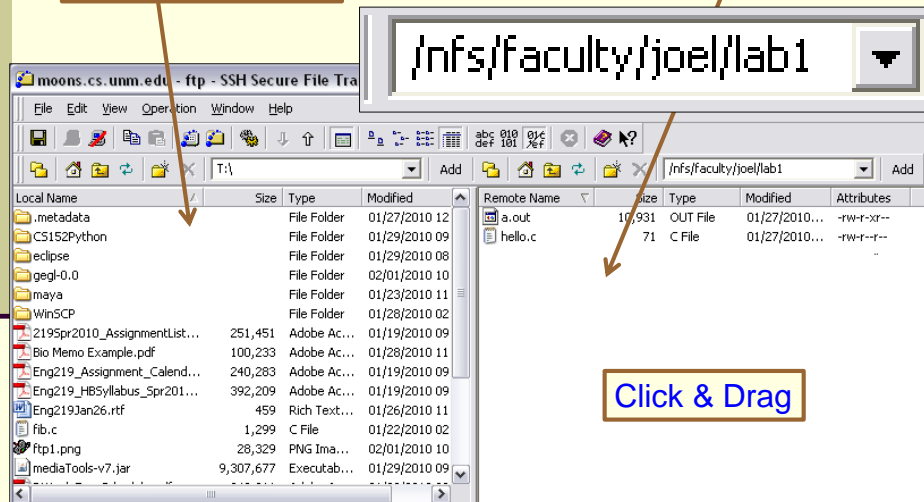
35

35

## UNM Labs: Moving files: SFTP: 2 of 2

Local File System

Remote File System



36

36

## How Do I Connect From My Machine?

**Question:** “At Home, I have a computer running  
△ OS version □. How do I telnet and ftp”?

**Answer:**



CS Support Group

[cssupport@cs.unm.edu](mailto:cssupport@cs.unm.edu)

Help Desk - 277-3527

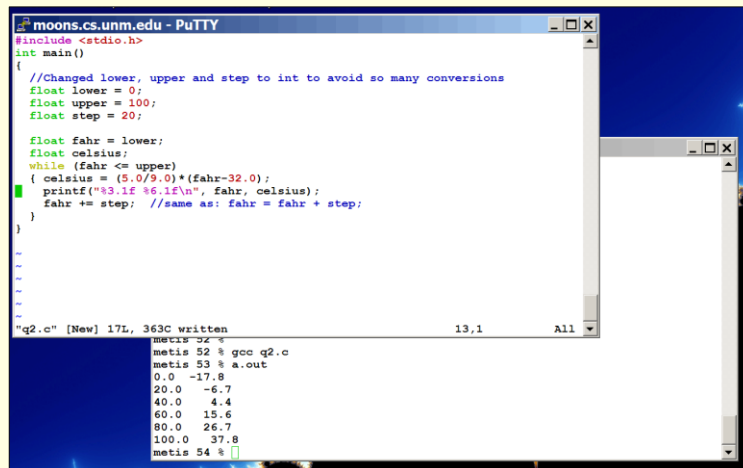
ECE 214A

37

37

## Workflow Suggestion: Duel PuTTY

Open two PuTTY window: one for editing and the other for  
compile and run.



```
moons.cs.unm.edu - PuTTY
#include <stdio.h>
int main()
{
    //Changed lower, upper and step to int to avoid so many conversions
    float lower = 0;
    float upper = 100;
    float step = 20;

    float fahr = lower;
    float celsius;
    while (fahr <= upper)
    {
        celsius = (5.0/9.0)*(fahr-32.0);
        printf("%3.1f %6.1f\n", fahr, celsius);
        fahr += step; //same as: fahr = fahr + step;
    }
}

"q2.c" [New] 17L, 363C written          13.1      All
metis ~x$
metis 52 $ gcc q2.c
metis 53 $ a.out
0.0  -17.8
20.0  -6.7
40.0   4.4
60.0  15.6
80.0  26.7
100.0 37.8
metis 54 $
```

38

38

## Power Linux

**<tab>** Auto complete a file name if enough letters have been typed to identify that file.

For example, if the only file in the current directory that starts with **gr** is **graphParser.c**, then **gr<tab>** will auto complete.

**history** Displays a history of the commands entered.

**↑** Walks back through the command history. Pressing **<enter>** on a selected command will repeat that command.

**!x** Repeats the most recent command starting with the sub-string *x*: **gcc graphParser.c**  
**!g**

39

39