CSCI2467: Systems Programming Concepts

Slide set 1: A Tour of Computer Systems (CS:APP Chapter 1)

Course Instructors:

Matthew Toups
Caitlin Boyce

Course Assistants:

Saroj Duwal
David McDonald

Spring 2020

THE UNIVERSITY of NEW ORLEANS

DEPARTMENT OF COMPUTER SCIENCE

Special wifi network in this classroom

- Sometimes the `unosecure` network gets overwhelmed
- Folks using laptops can end up with a slow network connection
- We have an alternative:

  wireless network name: `systems-lab`

  passphrase (includes spaces): `we love UNO CSCI`
- This will not work elsewhere on campus, but you may use this in here for the rest of the semester

Course notes

- Always monitor class schedule at: `http://2467.cs.uno.edu`
- No moodle for this course! We will use AutoLab

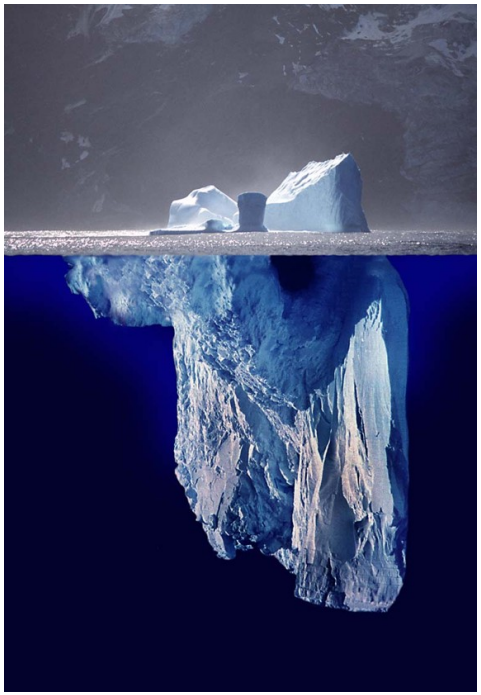
  `https://autolab.cs.uno.edu`
- Everyone downloaded `introlab-handout.tar` from AutoLab
- note difference between `systems-lab` and your own computer
- how to connect with ssh
- optional: set up a text editor to sync up automatically (Notepad++, Sublime Text, etc)

  See resources section of 2467 website for more info

Overview

- Course notes
1. A Tour of Computer Systems
   - Systems
   - Information is bits plus context
   - Programs Are Translated by Other Programs into Different Forms
   - Processors Read and Interpret Instructions Stored in Memory
   - Caches & Memory Hierarchy
   - The Operating System manages processes, memory, and more
   - Wrapping up the intro
2. Lab 0
   - Prerequisites review
   - Logging in to computer lab terminals
   - Remote access with `ssh`
   - The terminal
   - Text editors
   - Part 1
   - Part 2
   - Part 3
   - Handin notes
3. Preview of next class
   - Reading
   - Data surprises!

Systems vs. Applications



by: Uwe Kils CC BY-SA 3.0

Applications software provides services to a user

Systems software provides a platform for applications and services to build upon

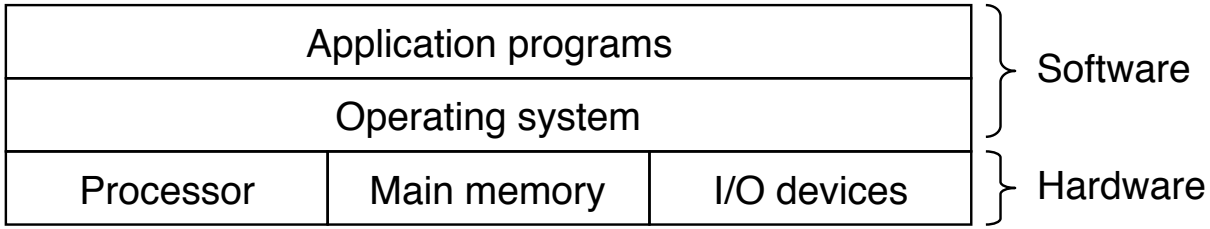
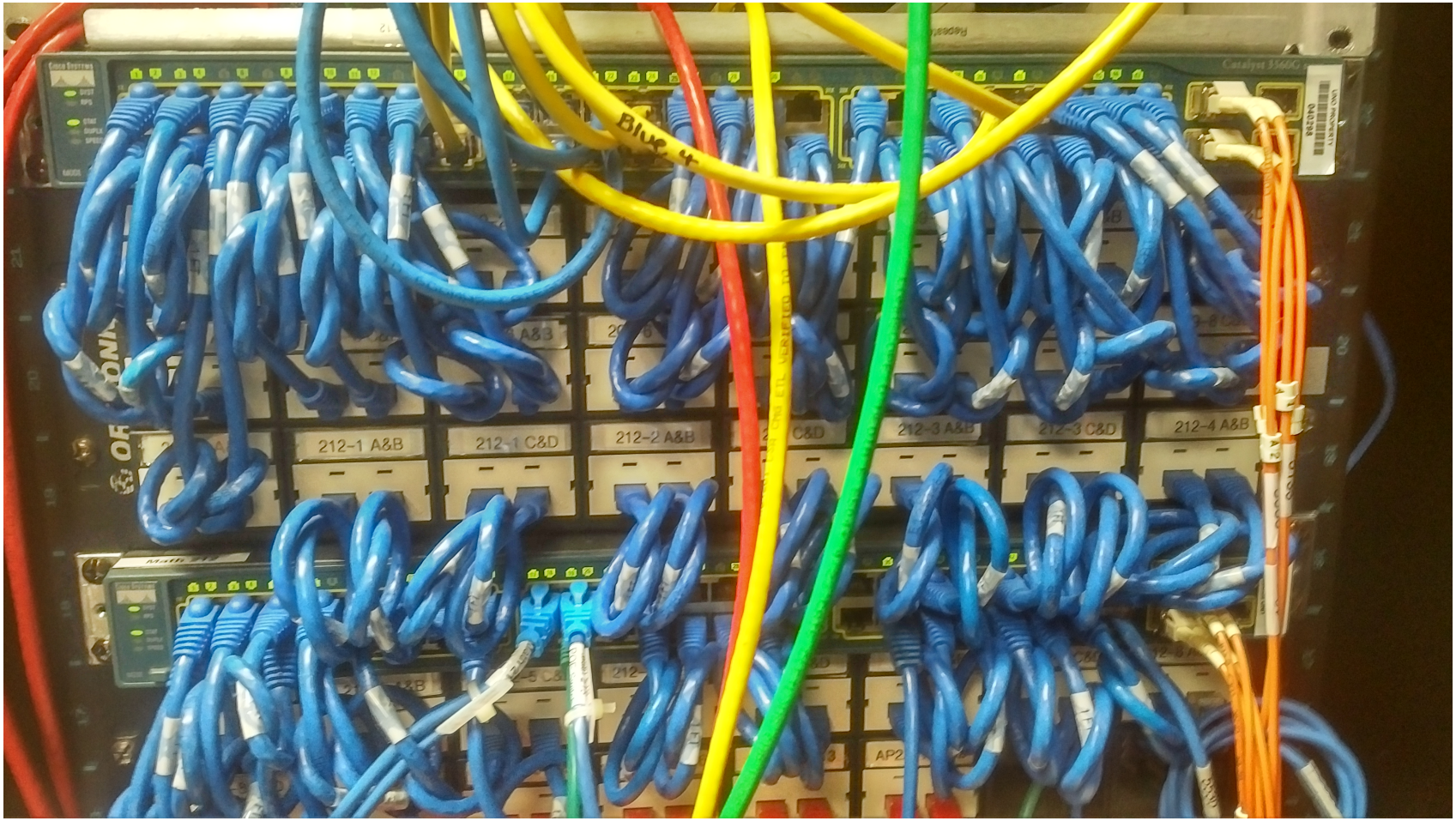


Figure 1.10 from CS:APP

Not pictured: numerous examples of libraries, firmware (for each hardware device, network switches), remote services

Systems code is everywhere
Running networks 24/7

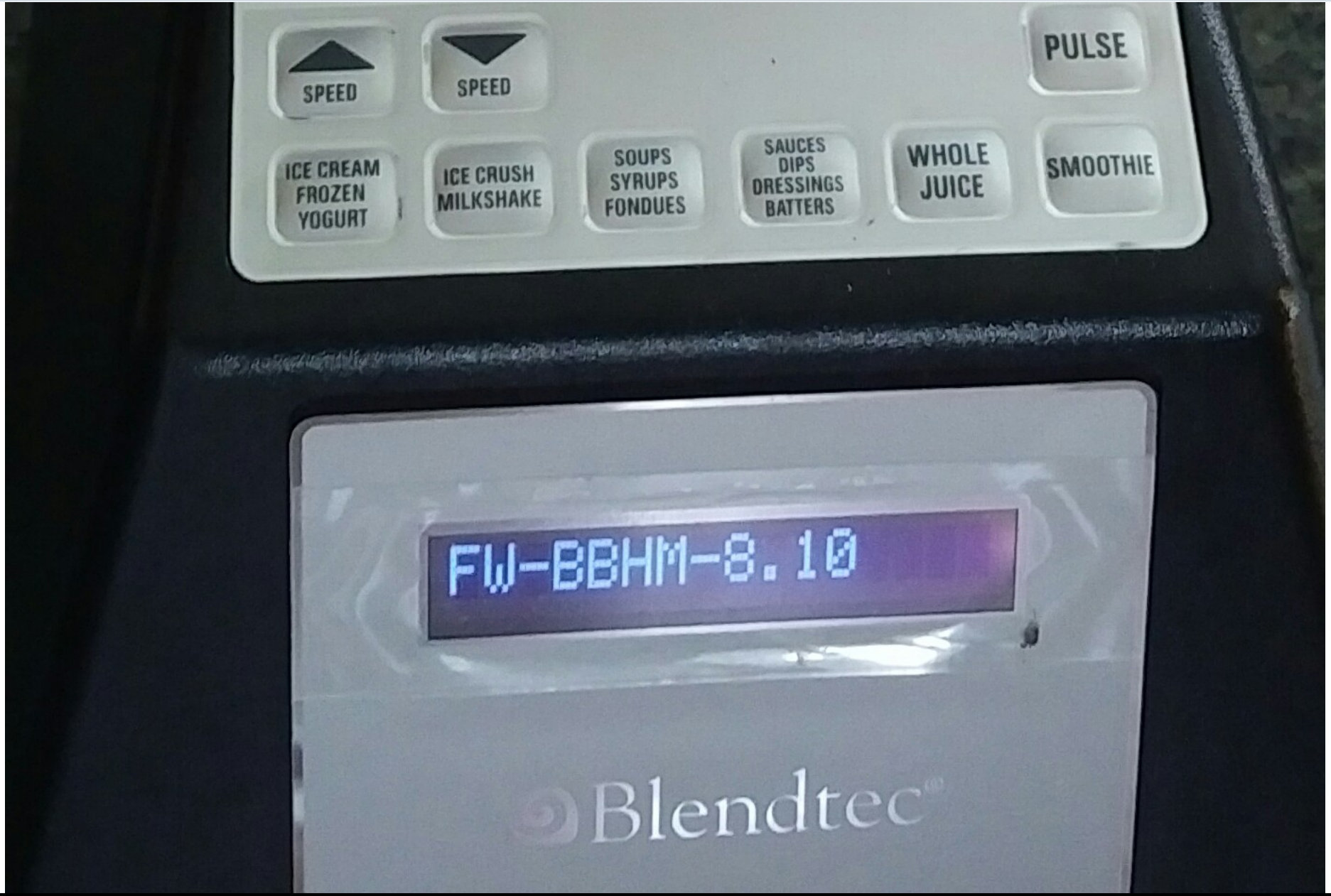
Cisco Catalyst network switch



Linksys WRT54G switch/router

Systems code is everywhere

Feeding us!



Creating a programming language for systems

Originally all systems level code written as assembly

- C developed 1969-1973 by Dennis Ritchie[1]
  - 1978: *K&R* book published
  - 1988: 2nd edition (ANSI/ISO standard)
  - C89/C90 most common. Some updates since then: C99, C11[2]
- C was created for a specific purpose: to implement the UNIX OS (kernel, libraries, tools)
- Small, simple language. Easily "ported" to different hardware.
- C becomes de facto choice for systems (OS, drivers, tools)
- C also becomes used in applications, but with some difficulty (pointers, fewer abstractions)

[1] See Aside on p.4 of CS:APP

[2] See Aside on p.35 of CS:APP

C: the Systems programming language

- No classes or objects
- (therefore no constructors or destructors)
- No exceptions (no `try` or `throw` or `catch`)
- No templates/generics
- ... C++ adds these things
- ..... making it very different! Don't confuse C and C++

code/intro/hello.c

```c
#include <stdio.h>

int main()
{
    printf("hello, world\n");
}
```

code/intro/hello.c

Figure 1.1: **The `hello` program.**

#	i	n	c	l	u	d	e	<sp>	<	s	t	d	i	o	.
35	105	110	99	108	117	100	101	32	60	115	116	100	105	111	46

h	>	\n	\n	i	n	t	<sp>	m	a	i	n	(	)	\n	{
104	62	10	10	105	110	116	32	109	97	105	110	40	41	10	123

\n	<sp>	<sp>	<sp>	<sp>	p	r	i	n	t	f	(	"	h	e	l
10	32	32	32	32	112	114	105	110	116	102	40	34	104	101	108

l	o	,	<sp>	w	o	r	l	d	\	n	"	)	;	\n	}
108	111	44	32	119	111	114	108	100	92	110	34	41	59	10	125

Figure 1.2: **The ASCII text representation of `hello.c`.**

hello.c (bits)

```
00100011 01101001 01101110 01100011 01101100 01110101  #inclu
01100100 01100101 00100000 00111100 01110011 01110100  de <st
01100100 01101001 01101111 00101110 01101000 00111110  dio.h>
00001010 00001010 01101001 01101110 01110100 00100000  ..int
01101101 01100001 01101001 01101110 00101000 00101001  main()
00001010 01111011 00001010 00100000 00100000 00100000  .{.
00100000 01110000 01110010 01101001 01101110 01110100   print
01100110 00101000 00100010 01101000 01100101 01101100  f("hel
01101100 01101111 00101100 00100000 01110111 01101111  lo, wo
01110010 01101100 01100100 01011100 01101110 00100010  rld\n"
00101001 00111011 00001010 00100000 00100000 00100000  );.
00100000 01110010 01100101 01110100 01110101 01110010   retur
01101110 00100000 00110000 00111011 00001010 01111101  n 0;.}
00001010                                               .
```

hello.c (hexadecimal)

```
2369 6e63 6c75 6465 203c 7374 6469 6f2e  #include <stdio.
683e 0a0a 696e 7420 6d61 696e 2829 0a7b  h>..int main().{
0a20 2020 2070 7269 6e74 6628 2268 656c  .    printf("hel
6c6f 2c20 776f 726c 645c 6e22 293b 0a20  lo, world\n");.
2020 2072 6574 7572 6e20 303b 0a7d 0a       return 0;.}.
```

hello.c (hexadecimal)

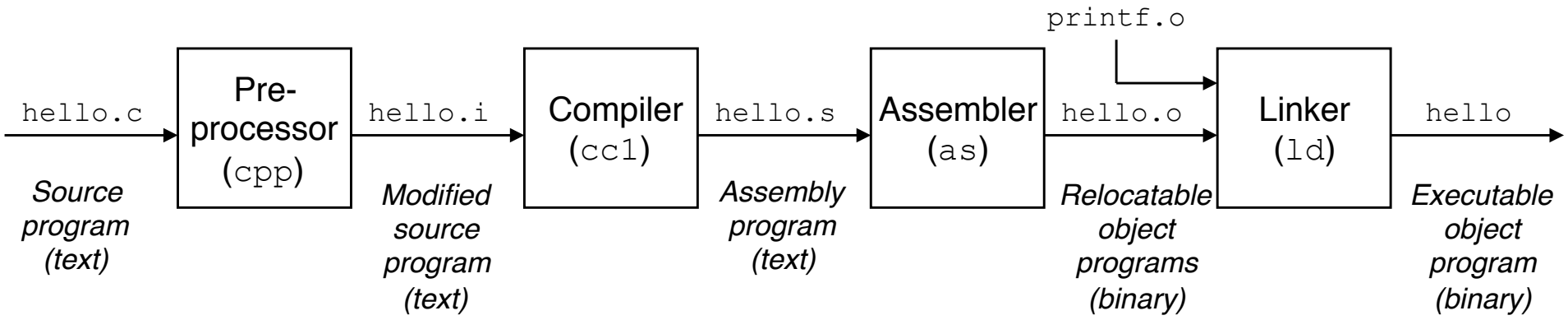
```
23 69 6E 63 6C 75 64 65 20 3C 73 74 64 69 6F 2E #include <stdio.23
68 3E 0A 0A 69 6E 74 20 6D 61 69 6E 28 29 0A 7B h>  int main()  {68
0A 20 20 20 20 70 72 69 6E 74 66 28 22 68 65 6C     printf("hel0A
6C 6F 2C 20 77 6F 72 6C 64 5C 6E 22 29 3B 0A 20 lo, world\n");  6C
20 20 20 72 65 74 75 72 6E 20 30 3B 0A 7D 0A        return 0;  }  20
```

`hello` compiled (hexadecimal)

```
7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00 |.ELF............
02 00 3E 00 01 00 00 00 30 04 40 00 00 00 00 00 |..>.....0.@.....
40 00 00 00 00 00 00 00 E0 19 00 00 00 00 00 00 |@...............
00 00 00 00 40 00 38 00 09 00 40 00 1F 00 1C 00 |....@.8...@.....
06 00 00 00 05 00 00 00 40 00 00 00 00 00 00 00 |........@.......
40 00 40 00 00 00 00 00 40 00 40 00 00 00 00 00 |@.@.....@.@.....
F8 01 00 00 00 00 00 00 F8 01 00 00 00 00 00 00 |................
08 00 00 00 00 00 00 00 03 00 00 00 04 00 00 00 |................
38 02 00 00 00 00 00 00 38 02 40 00 00 00 00 00 |8.......8.@.....
38 02 40 00 00 00 00 00 1C 00 00 00 00 00 00 00 |8.@.............
1C 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 |................
01 00 00 00 05 00 00 00 00 00 00 00 00 00 00 00 |................
00 00 40 00 00 00 00 00 00 00 40 00 00 00 00 00 |..@.......@.....4C 89 EA 4C
FC 06 00 00 00 00 00 00 FC 06 00 00 00 00 00 00 |................01 48 39 EB
00 00 20 00 00 00 00 00 01 00 00 00 06 00 00 00 |.. .............41 5E 41 5F
10 0E 00 00 00 00 00 00 10 0E 60 00 00 00 00 00 |..........`.....F3 C3 00 00
10 0E 60 00 00 00 00 00 28 02 00 00 00 00 00 00 |..`.....(.......01 00 02 00
30 02 00 00 00 00 00 00 00 00 20 00 00 00 00 00 |0......... .....00 00 00 00
02 00 00 00 06 00 00 00 28 0E 00 00 00 00 00 00 |........(.......1C FE FF FF
28 0E 60 00 00 00 00 00 28 0E 60 00 00 00 00 00 |(.`.....(.`.....52 FF FF FF
D0 01 00 00 00 00 00 00 D0 01 00 00 00 00 00 00 |................DC FF FF FF
08 00 00 00 00 00 00 00 04 00 00 00 04 00 00 00 |................01 7A 52 00
54 02 00 00 00 00 00 00 54 02 40 00 00 00 00 00 |T.......T.@.....14 00 00 00
```

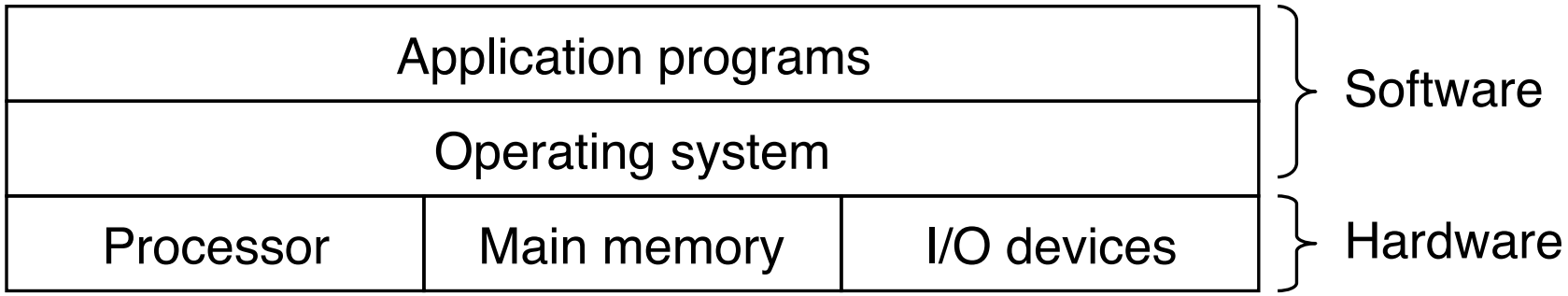
Overview

Steps in which `hello.c` gets converted to an executable



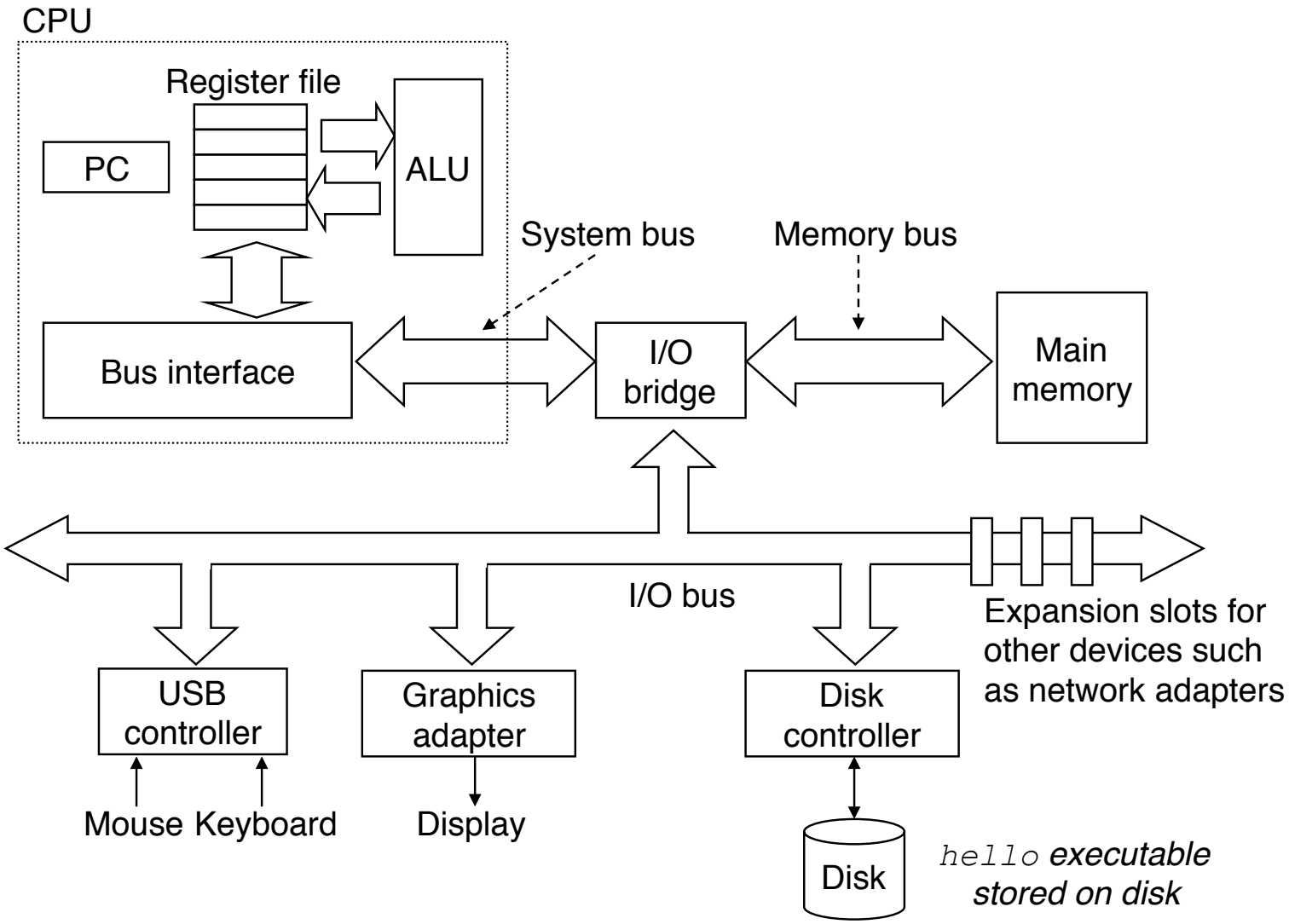
Source: CS:APP, Bryant & O'Hallaron

Layered view of systems



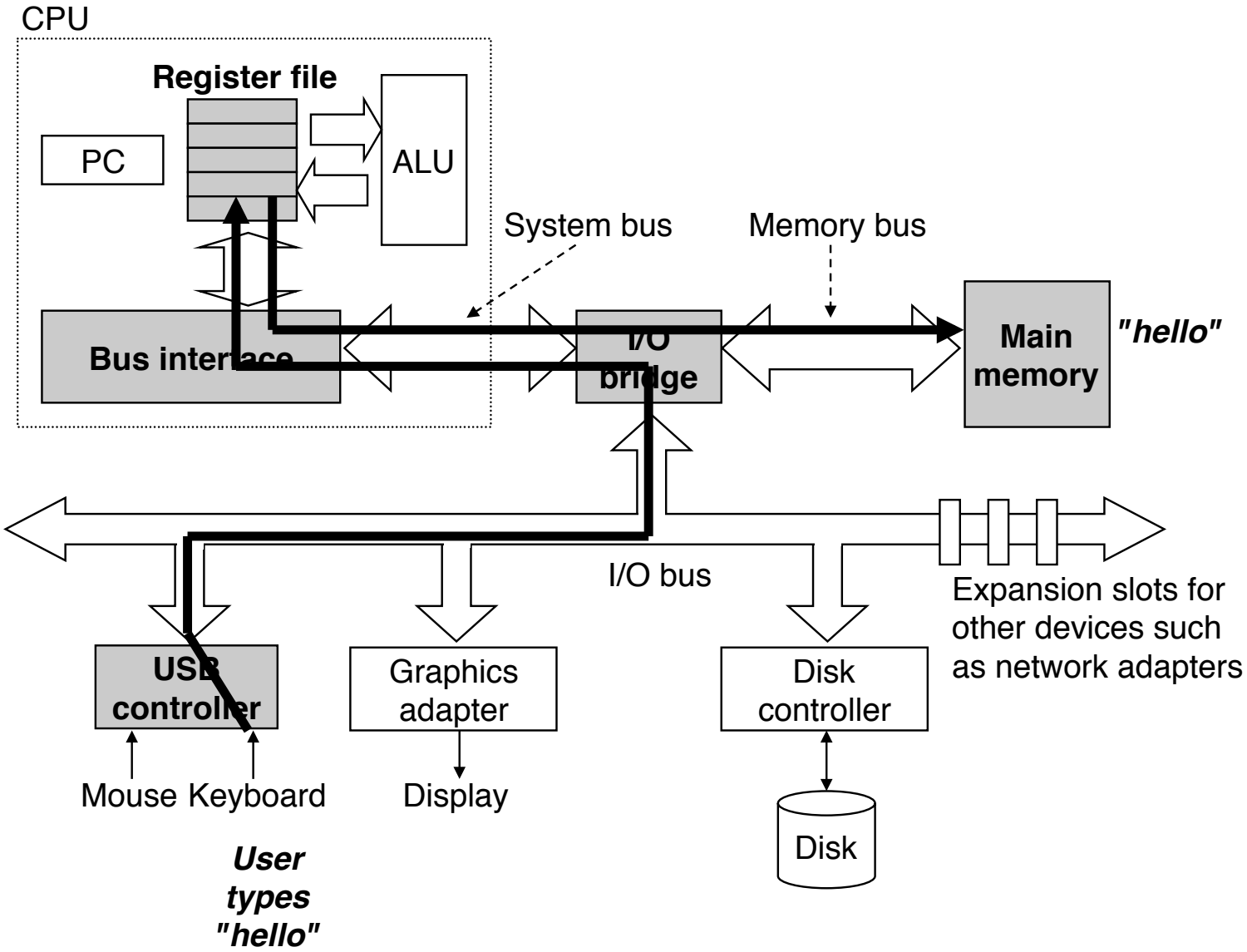
Source: CS:APP, Bryant & O'Hallaron

A diagram of the computer's hardware



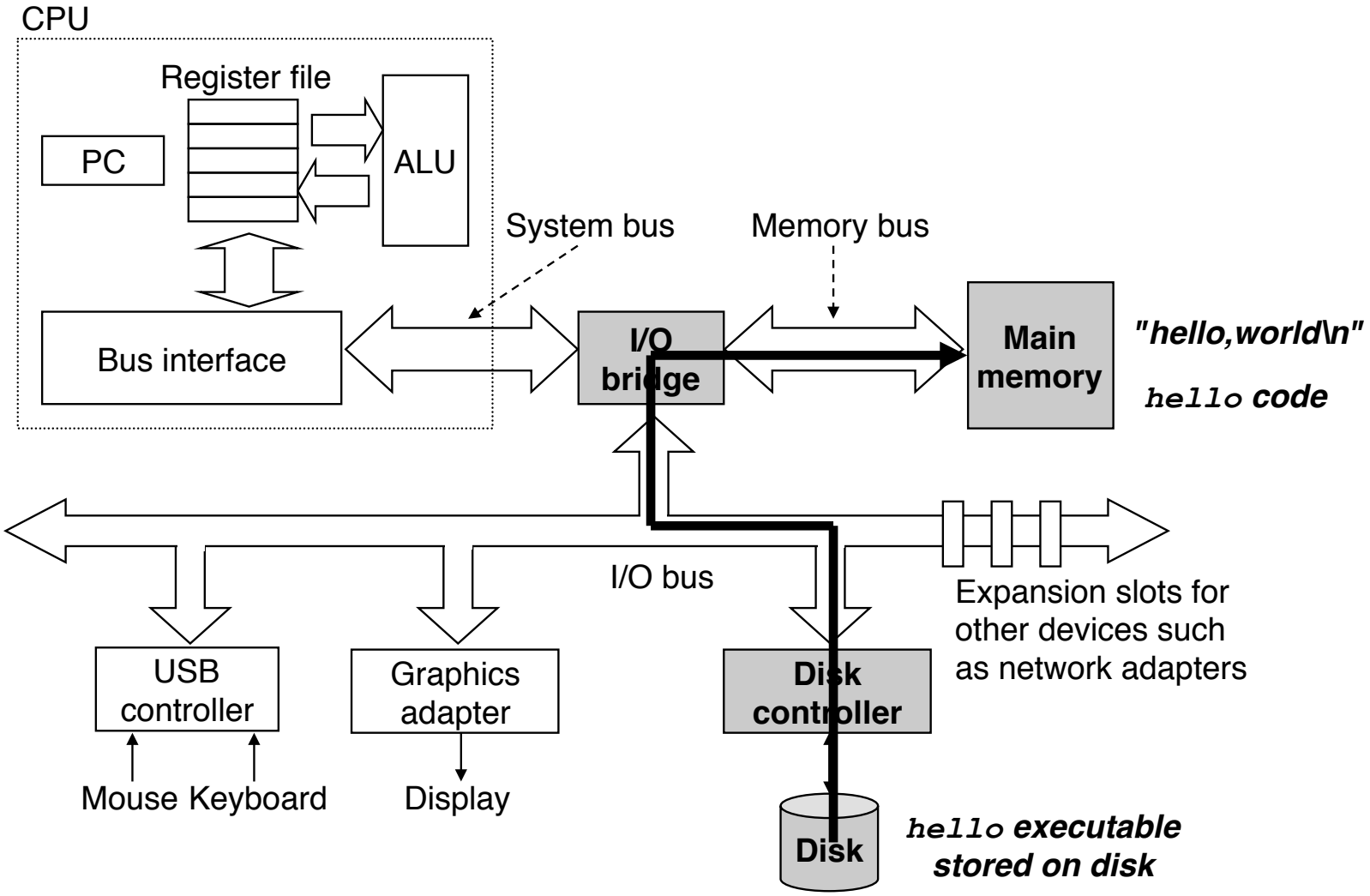
Source: CS:APP, Bryant & O'Hallaron

Someone types in the `hello` command...



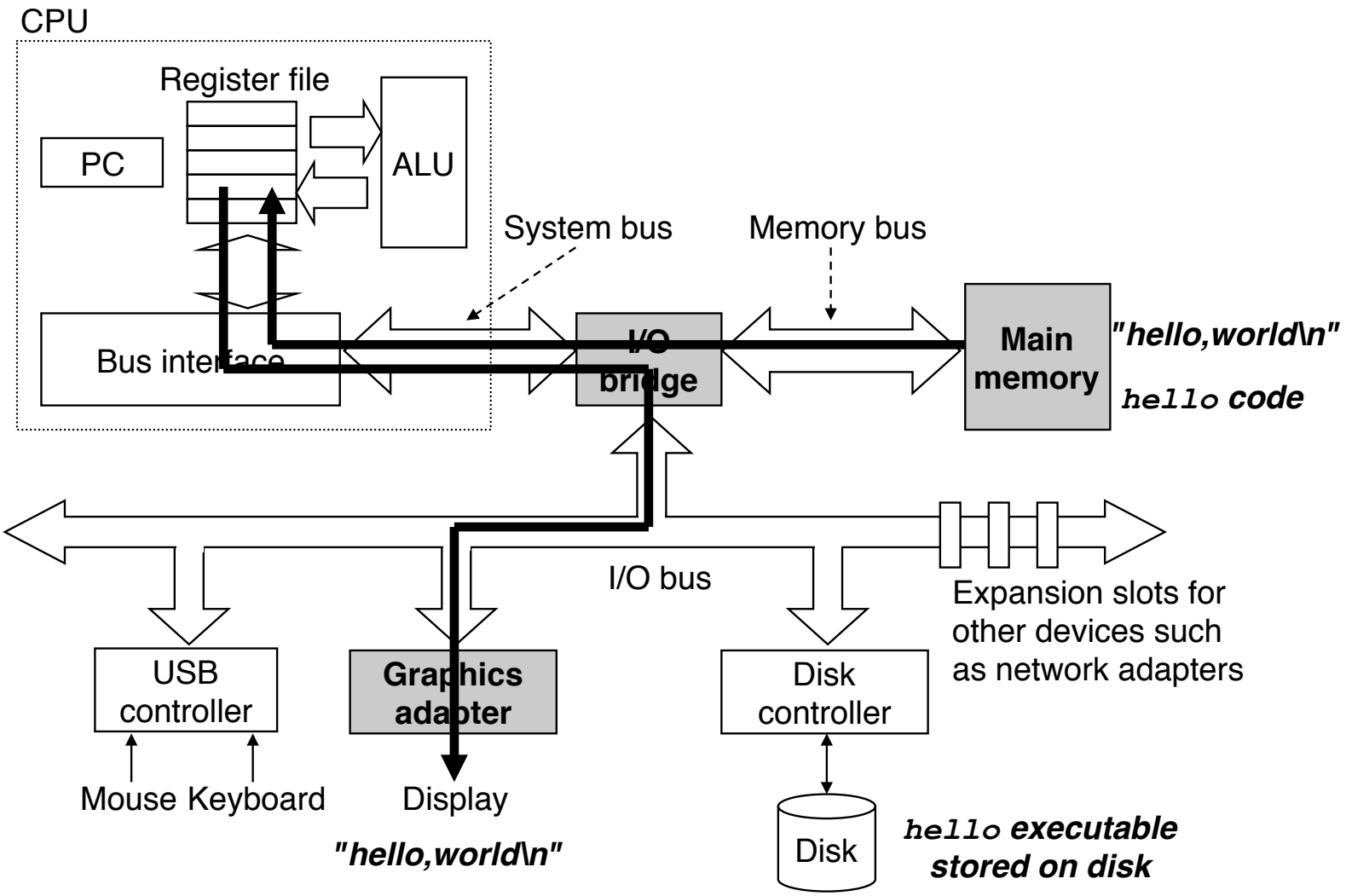
Source: CS:APP, Bryant & O'Hallaron

OS loads `hello` from the disk into memory



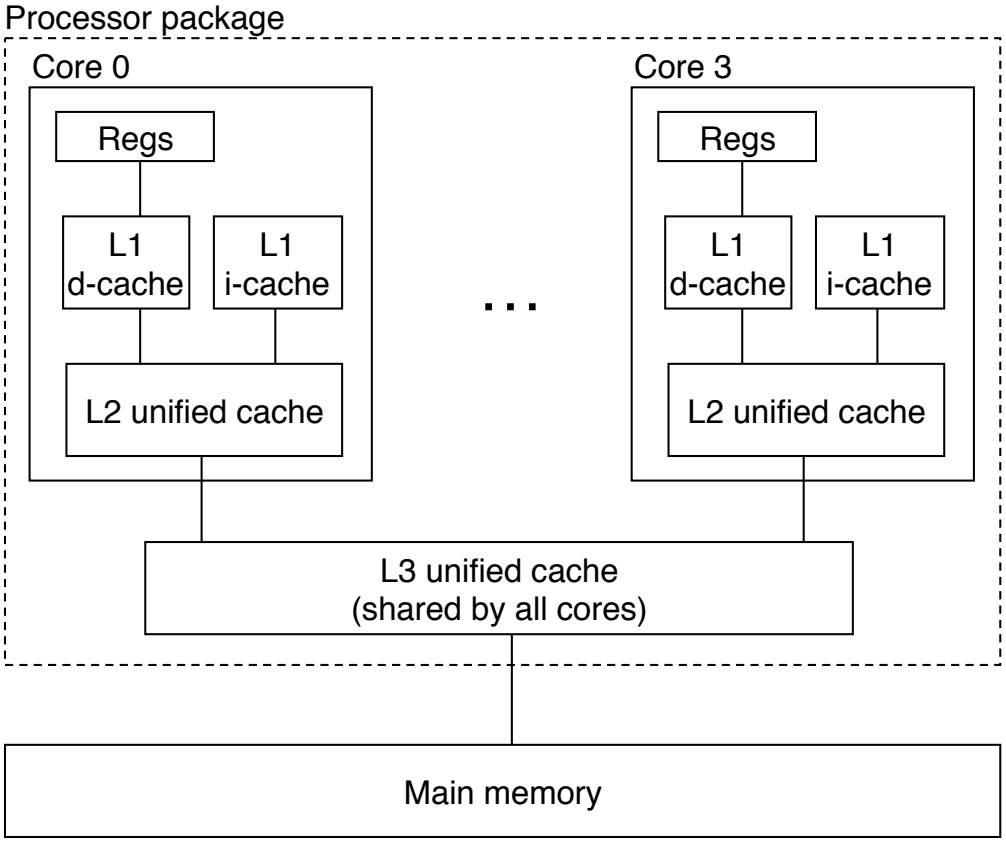
Source: CS:APP, Bryant & O'Hallaron

How `hello` output reaches the display

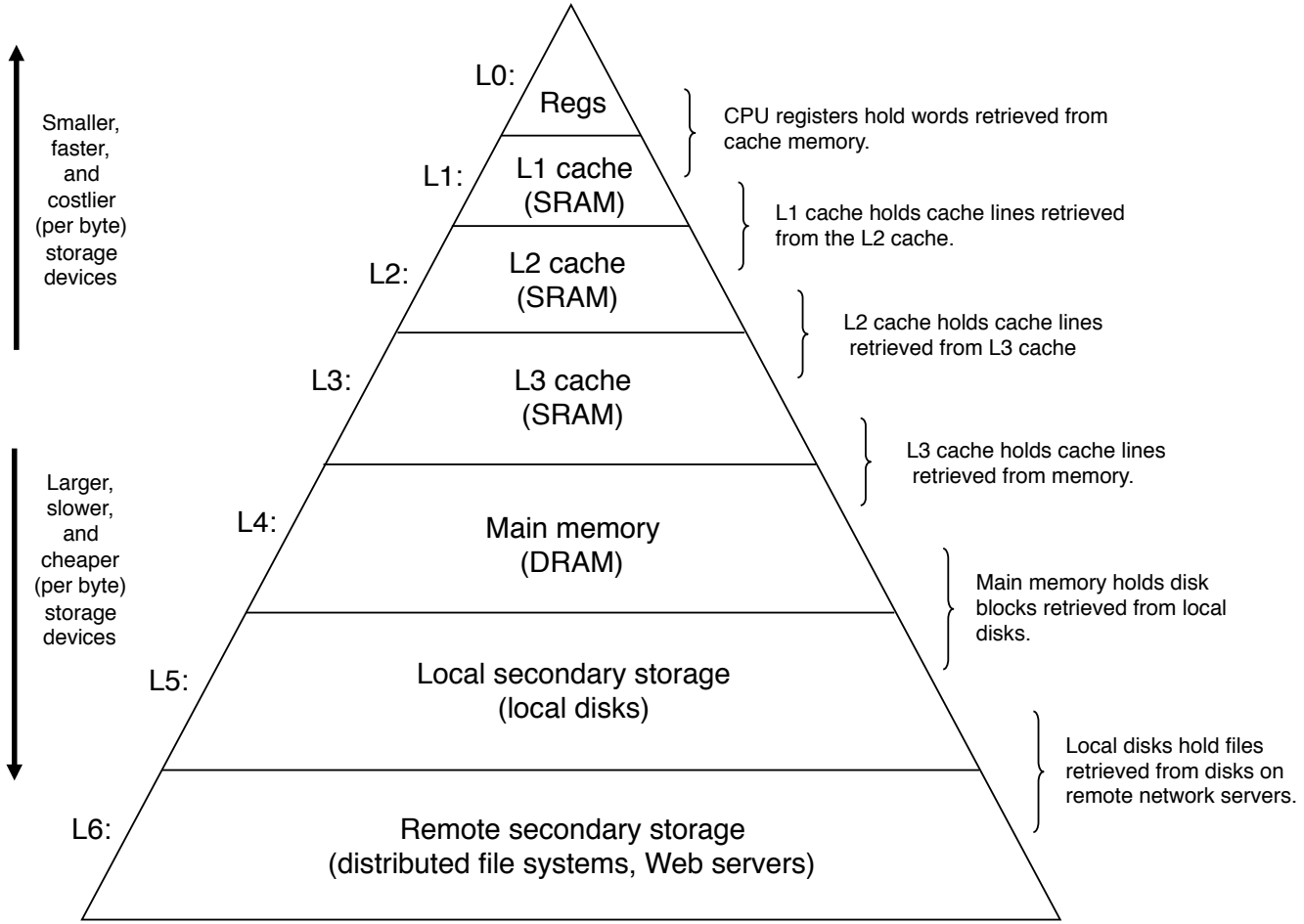


Source: CS:APP, Bryant & O'Hallaron

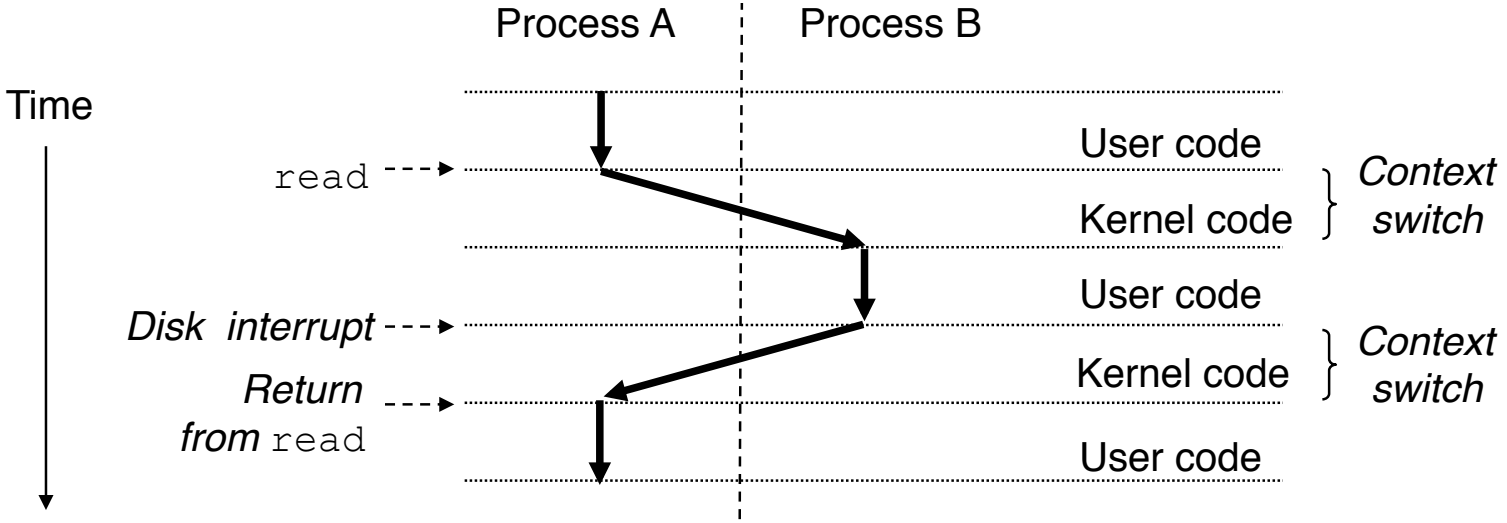
Caches on the Intel Core i7 CPU



Memory Hierarchy: many layers, from fast down to slow



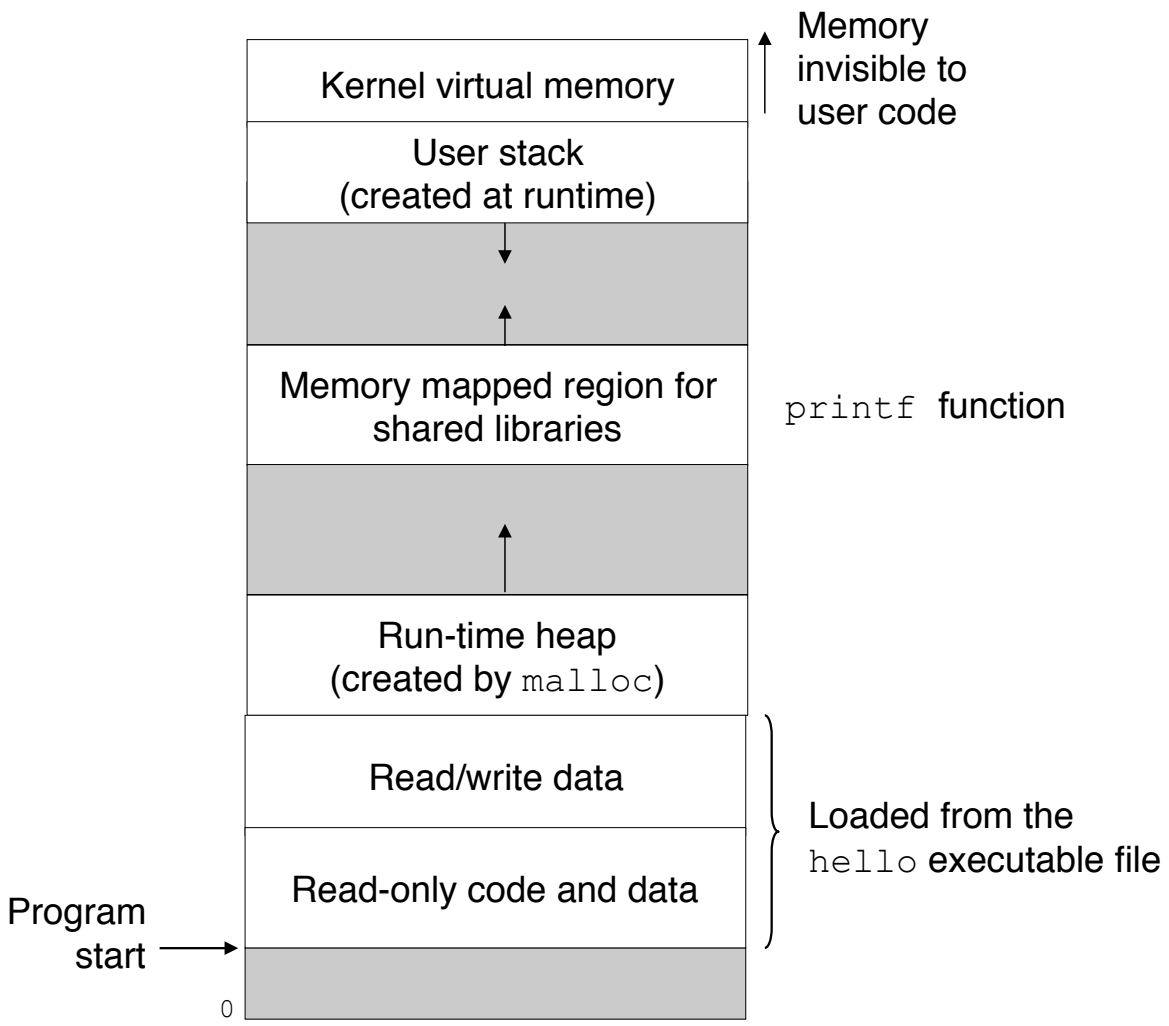
OS Kernel handles switching between two processes



Source:

CS:APP, Bryant & O'Hallaron

OS Kernel also manages Virtual Memory addressing



Source: CS:APP, Bryant & O'Hallaron

Chapter 1: to sum up

- Systems skills give you a broad and deep insight into what happens when you write and/or run programs
- Potentially big impacts on speed, reliability, security

- This is a brief preview of what is to come! You aren't expected to know all of this ... yet
- Chapter 1 should be a quick read.
- Don't worry about memorizing Amdahl's law, it is a useful principle to keep in mind but I will not ask you to solve those math problems

2. Lab 0
   - Prerequisites review
   - Logging in to computer lab terminals
   - Remote access with `ssh`
   - The terminal
   - Text editors
   - Part 1
   - Part 2
   - Part 3
   - Handin notes

Your first lab assignment

THE UNIVERSITY of NEW ORLEANS
DEPARTMENT OF COMPUTER SCIENCE

CSCI 2467, Spring 2020

Lab 0: Introductions to C and Unix

Due: Wednesday, January 22, 11:59PM

2467 Instructors: M. Toups & C. Boyce
Assistants: D. McDonald & S. Duwal
staff@2467.cs.uno.edu

1 Introductions

The purpose of this assignment is to perform two introductions while getting you started in your exploration of Computers from a Systems perspective. Chapter 1 of Bryant & O'Hallaron's CS:APP textbook will accompany this lab conceptually, and the tools we use will be explained further in the two PDF documents from Stanford's CS Library (*Essential C* and *Unix Programming Tools*).

Please keep both the textbook and these two supplements handy as we work through this lab. The

Prerequisites

A few notes on this process: bash shell, text editors, permissions

Also: remote access via `ssh`

Be careful about usernames, and be aware there is a temporary block after a certain number of failed login attempts.

Using class server: `systems-lab`

Log in using UNO username, password.

All labs are done using `systems-lab` (linux) ...

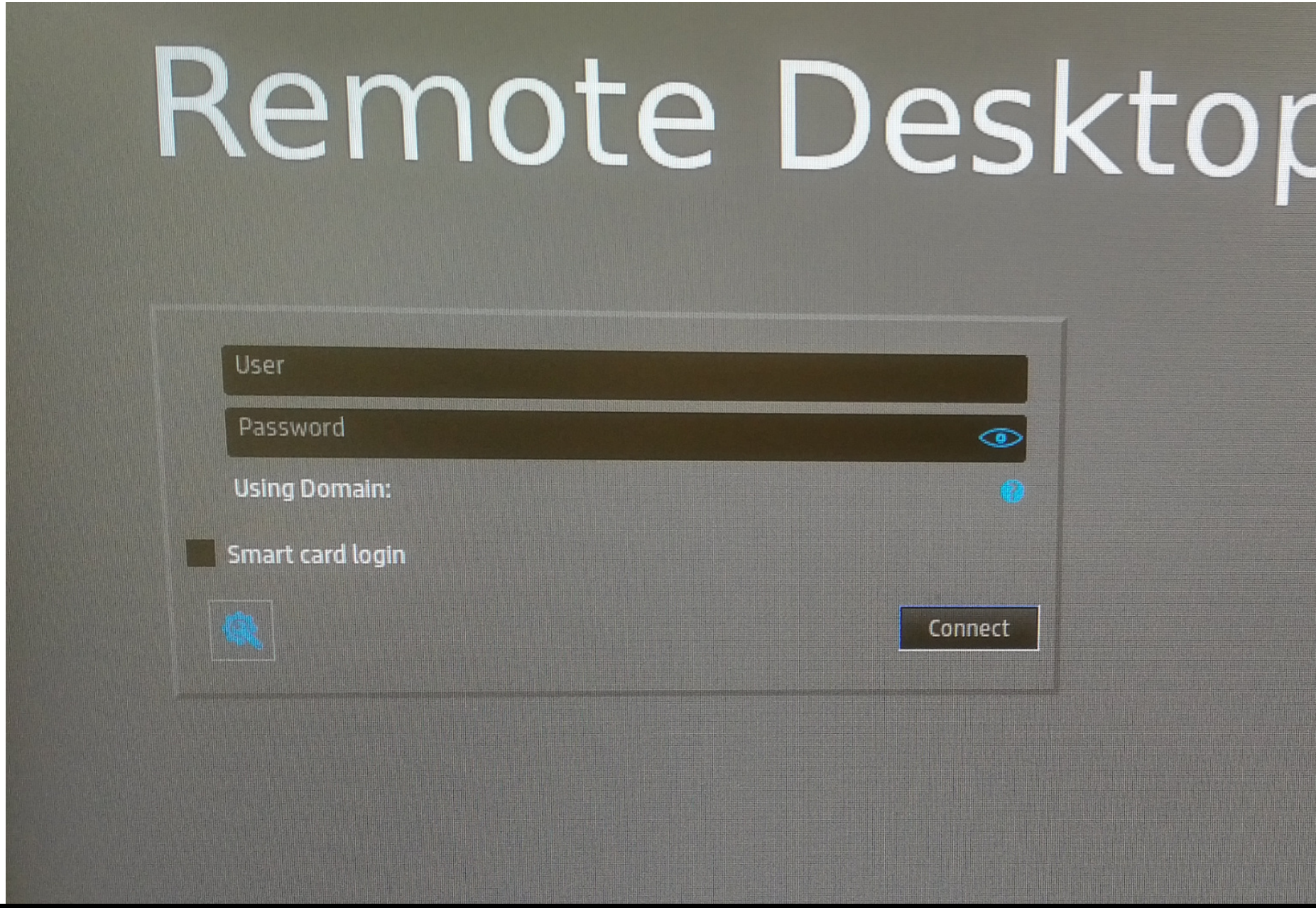
BUT

you can access `systems-lab` on the HP terminals in 209/212

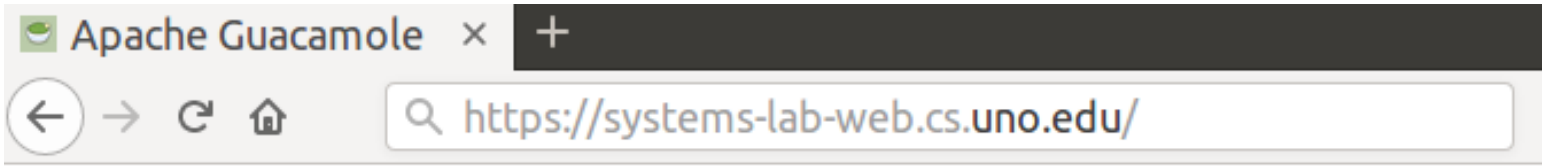
OR

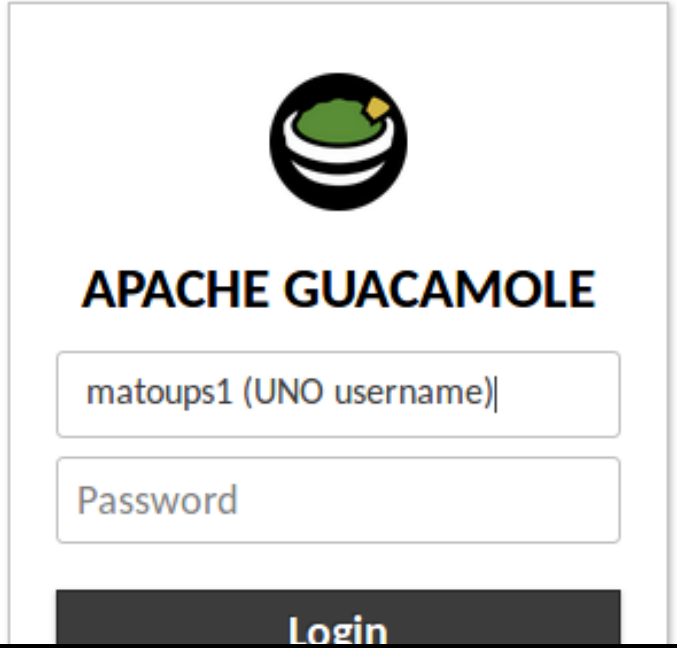
remotely (on or off-campus) using `ssh`

Remote Desktop login on HP thin client



Alternative on your own computer: systems-lab-web.cs.uno.edu





Using *git-bash* and ssh

Pay attention to which prompt is which

(Note: replace `matoups2` with your UNO username)

MINGW64:/i

```
UNO-DOMAIN+matoups2@math327-vm8 MINGW64 /i
$ ssh matoups2@systems-lab.cs.uno.edu
```

Using ssh from macOS

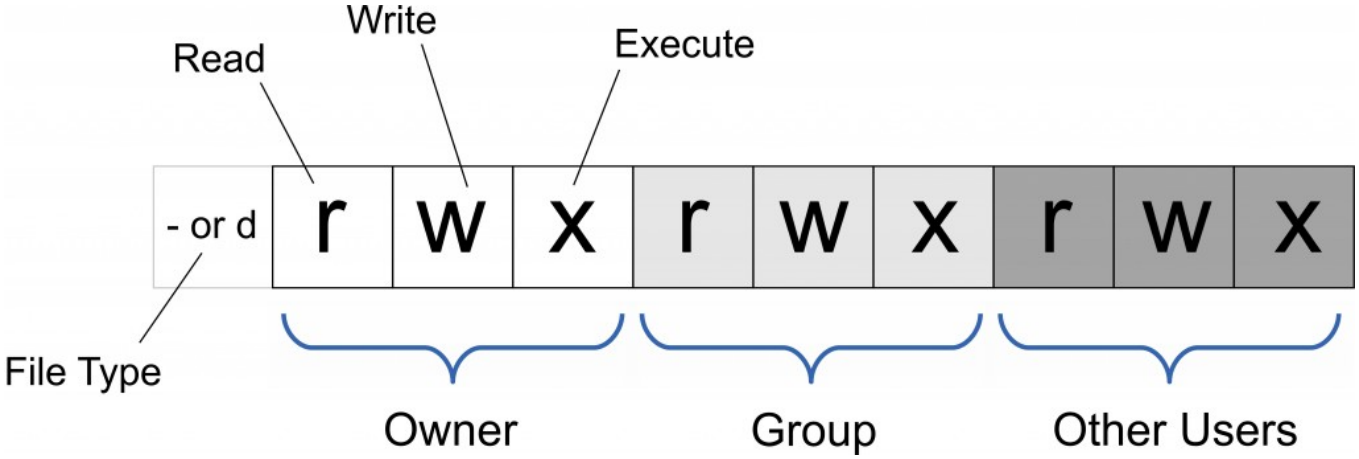
Optional: adding RSA key for login

```
ssh
davidmcdonald@Davids-MacBook-Pro ~ % ssh dgmcdona@systems-lab.cs.uno.edu
dgmcdona@systems-lab.cs.uno.edu's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-74-generic x86_64)
```

```
ssh
davidmcdonald@Davids-MacBook-Pro ~ % ssh dgmcdona@systems-lab.cs.uno.edu
dgmcdona@systems-lab.cs.uno.edu's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-74-generic x86_64)

  System information as of Thu Jan 16 21:14:51 CST 2020

  System load:  0.27                Users logged in:     2
  Usage of /:   76.8% of 31.37GB    IP address for eth0: 172.16.20.2
  Memory usage: 18%                 IP address for eth1: 192.168.1.31
  Swap usage:   0%                  IP address for eth2: 137.30.120.31
  Processes:    3168

Last login: Thu Jan 16 21:13:04 2020 from 196.247.50.124
dgmcdona@systems-lab:~$ mkdir .ssh
dgmcdona@systems-lab:~$ touch .ssh/authorized_keys
dgmcdona@systems-lab:~$ cat id_rsa.pub >> ~/.ssh/authorized_keys
dgmcdona@systems-lab:~$ rm id_rsa.pub
dgmcdona@systems-lab:~$ exit
logout
Connection to systems-lab.cs.uno.edu closed.
```

Directories and permissions

- created a directory with `mkdir` command
- moved a file with the `mv` command
- changed directory with `cd` command
- changed permissions with `chmod` command



Directories and permissions

permissions		user	group	size	date	file/directory
drwxr-xr-x	2	paul	users	1024	Jan 2 23:50	.
drwxr-xr-x	6	root	root	1024	Jan 2 22:51	..
drwxr-xr-x	3	paul	users	1024	Jan 8 11:42	grassdata
lrwxrwxrwx	1	paul	users	13	May 6 1998	latex -> /d2/lt
drwx------	2	paul	users	1024	Mar 8 17:30	mail
drwx------	2	paul	users	1024	Feb 4 01:09	projects
-rw-r--r--	1	paul	users	844344	Dec 9 1998	nations.ps
-rw-rw-r--	1	paul	users	21438	Mar 2 21:47	ps4mf.txt

other (world) permissions
group permissions
user permissions

r : read permission
w : write permission
x : execute permission (programm)
– : permission not set

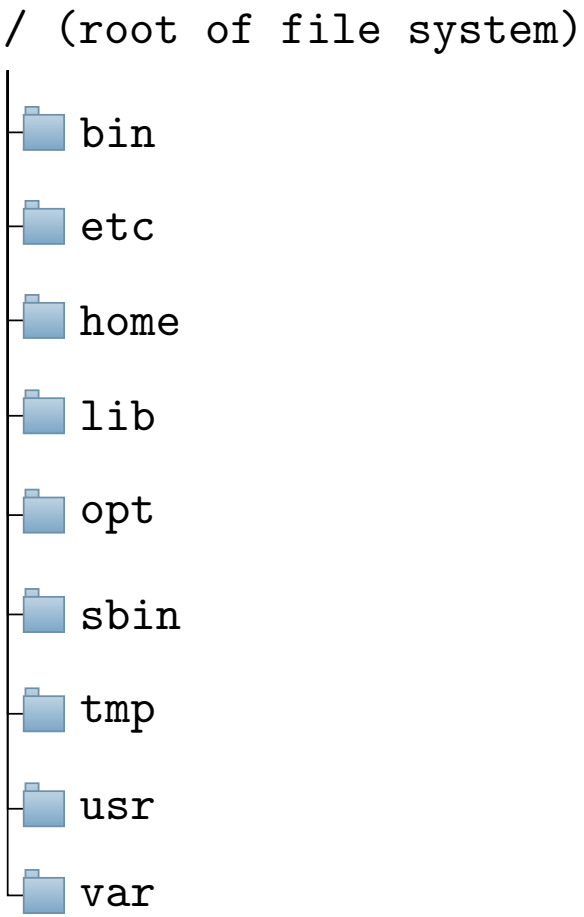
d : directory
– : file
l : link (to other file/directory)

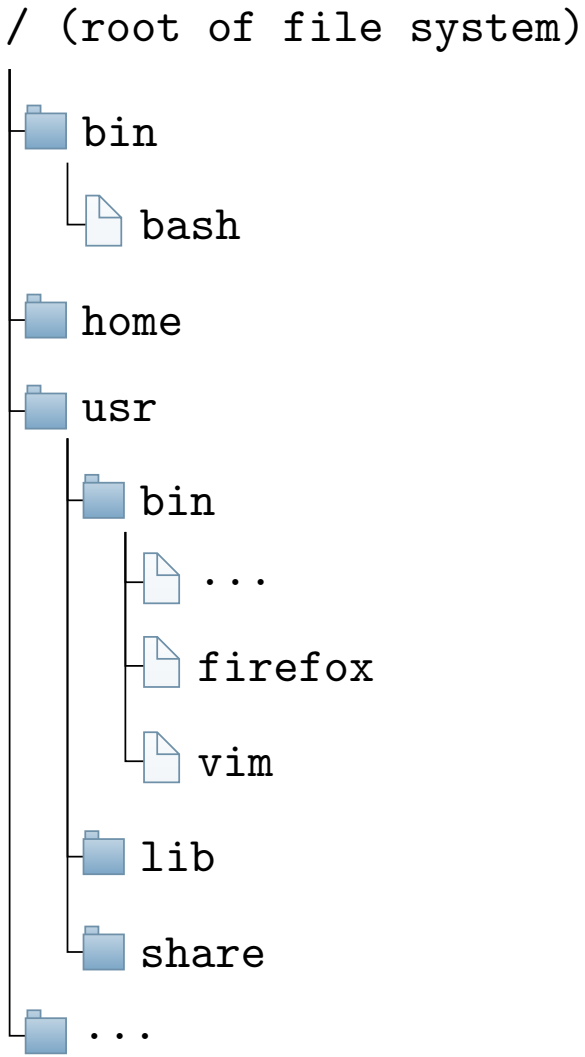
Source:

Open Source GIS: A GRASS GIS Approach, First Edition, 2002

Directory hierarchy

... and files within that directory





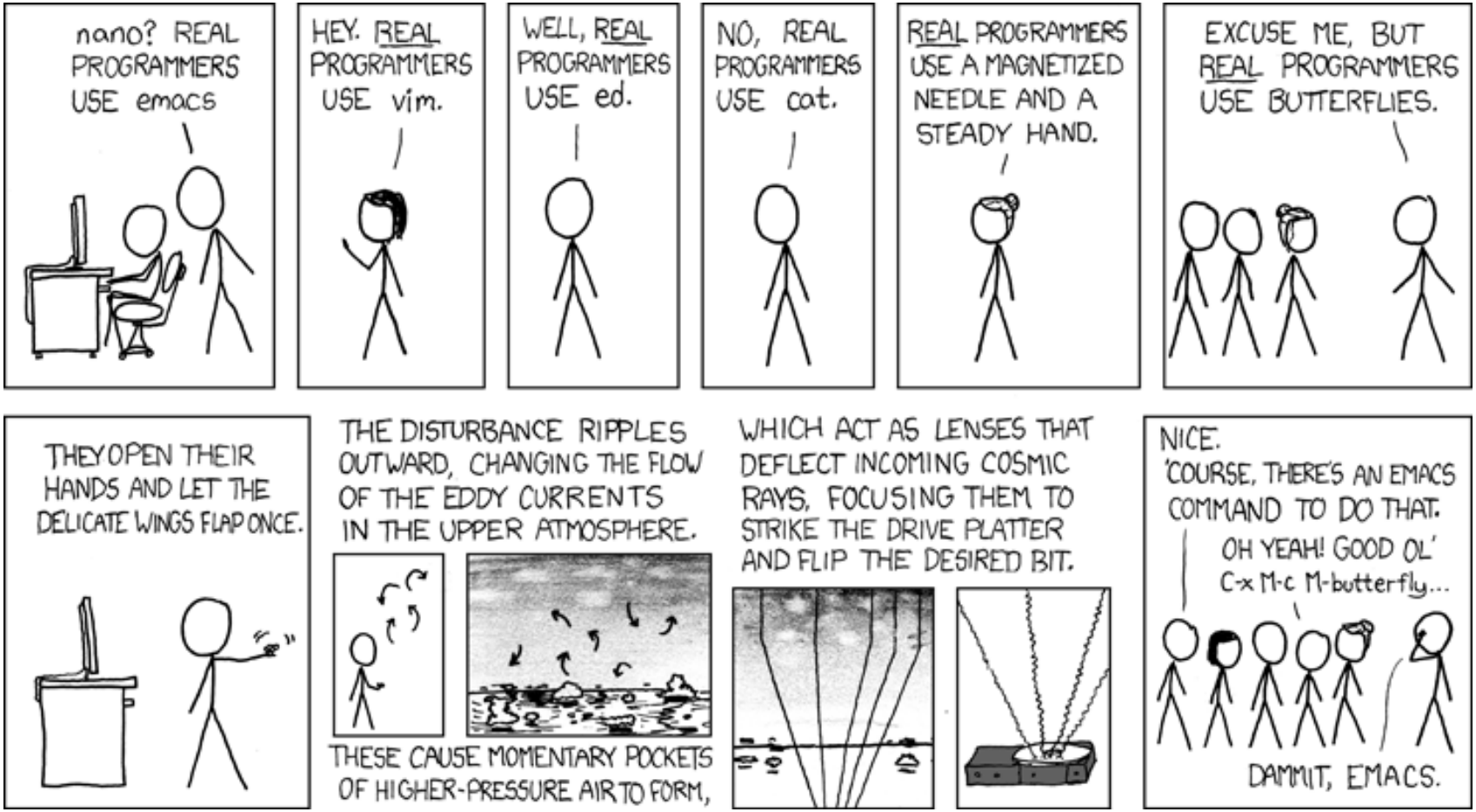
Terminal Shortcuts (bash shell)

- ~ is an alias to your home directory:
  `cp foo.txt ~`
  equivalant to: `cp foo.txt /home/matoups2/`
- . is an alias to your present directory:
  `cp ~/foo.txt .`
- .. is an alias to the parent directory:
  `cp ~/foo.txt ..`
- * will match as many characters as it can.
  `cp ~/*.txt .`
- *Example:* `objdump -d *`
- *Example:* rm *.c (be very very very careful!!)
- **There is no "trash" with `rm`. Your file will be gone.**

More bash tricks

- Pressing tab will autocomplete filenames.
- Use the up & down arrow keys to scroll through your previous commands.
- Control+R lets you search your command history.
- Control+C terminates your current program.
- Control+D (on a blank line) exits the terminal.

Text editors



Source: xkcd.com

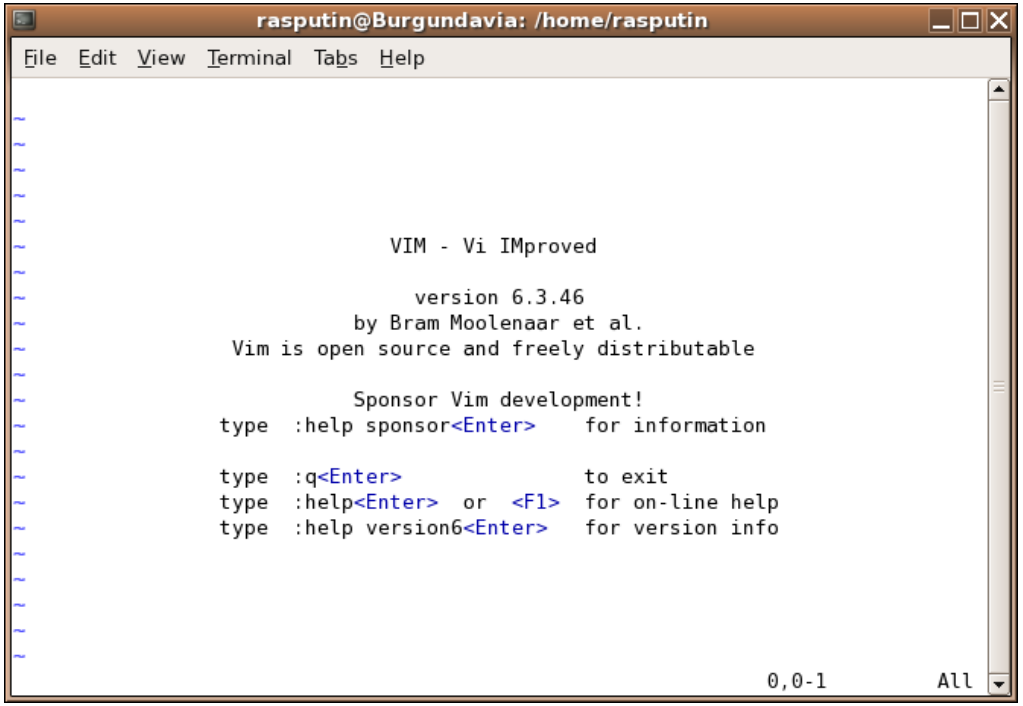
nano

If you aren't sure where to start, use nano first. You can easily finish *introlab* with nano even if you have not used it before.

```
  GNU nano 2.2.6          File: hello.c

#include <stdio.h>
#define RETURNVALUE 67

int main()
{
    int retval = RETURNVALUE;

    printf("Hello world.\n");

    return retval;
}



^G Get Help^O WriteOut^R Read Fil^Y Prev Pag^K Cut Text^C Cur Pos
^X Exit    ^J Justify ^W Where Is^V Next Pag^U UnCut Te^T To Spell
```

Note command-key functions described at bottom

vi / vim



see `vimtutor` command

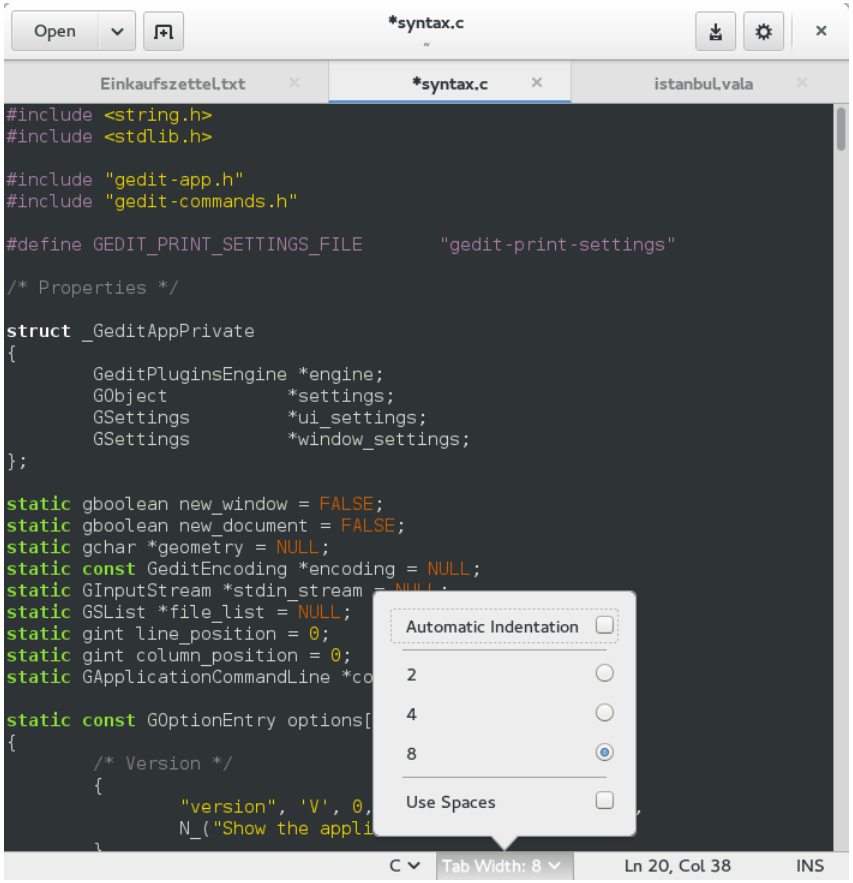
"stateful" or "modal" editor: can be in *command*, *insert* or *visual* mode.

emacs

```
File Edit Options Buffers Tools Help
Welcome to GNU Emacs, one component of a Linux-based GNU system.

Get help           C-h  (Hold down CTRL and press h)
Undo changes       C-x u       Exit Emacs              C-x C-c
Get a tutorial     C-h t       Use Info to read docs   C-h i
Ordering manuals   C-h RET
Activate menubar   F10  or  ESC `  or   M-`
(`C-' means use the CTRL key.  `M-' means use the Meta (or Alt) key.
If you have no Meta key, you may instead type ESC followed by the character.)

GNU Emacs 21.4.1 (i486-pc-linux-gnu, X toolkit, Xaw3d scroll bars)
 of 2007-06-19 on ninsei, modified by Debian
Copyright (C) 2001 Free Software Foundation, Inc.

GNU Emacs comes with ABSOLUTELY NO WARRANTY; type C-h C-w for full details.
Emacs is Free Software--Free as in Freedom--so you can redistribute copies
of Emacs and modify it; type C-h C-c to see the conditions.
Type C-h C-d for information on getting the latest version.

--uu:---F1  *scratch*        (Lisp Interaction)--L1--All----------------------
For information about the GNU Project and its goals, type C-h C-p.
```

Described in Section 4 of *Unix programming tools* (optional reading)

gedit

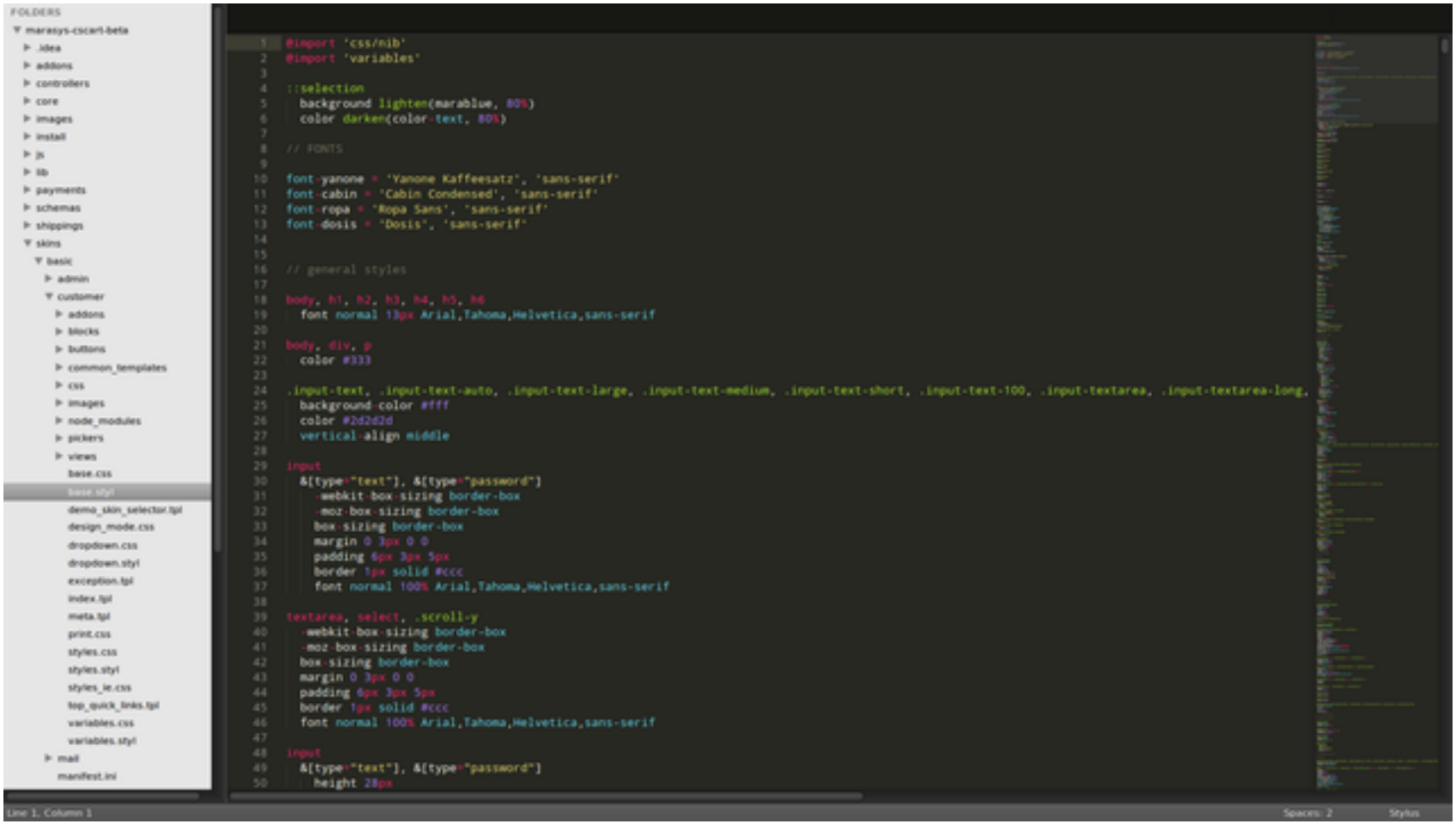


Nice GUI you can use in the lab

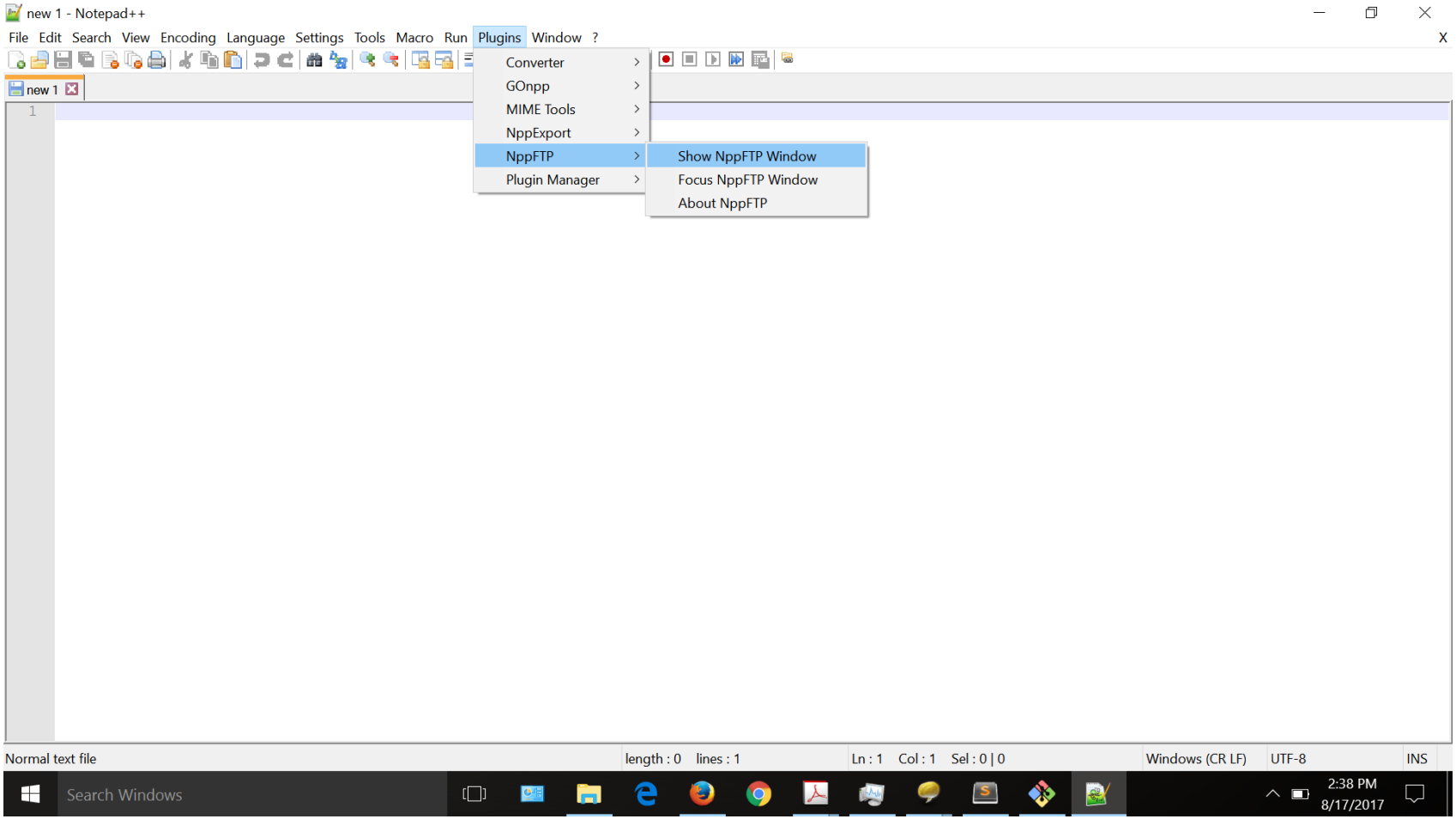
Disadvantage: doesn't work (easily) via ssh

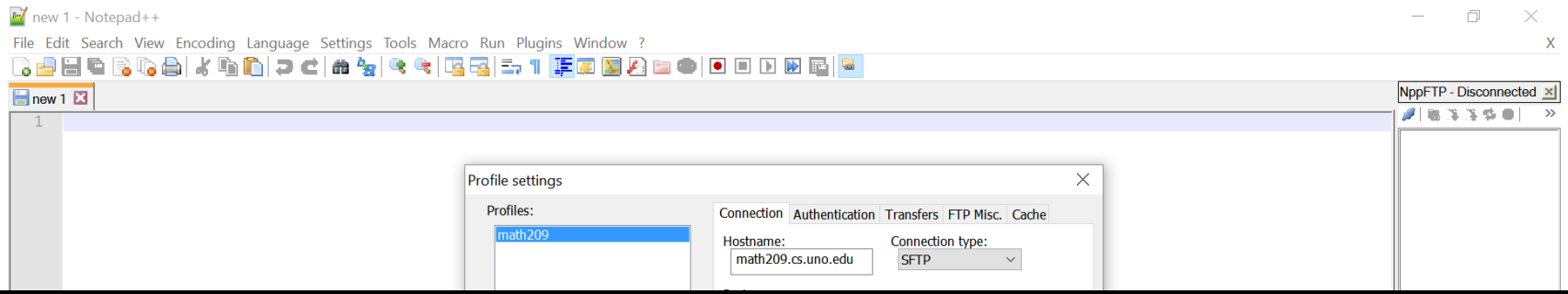
Previous examples were all *text-mode* (terminal-based, work the same over ssh)

Sublime Text



Notepad++


new 1 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
new 1
Converter
GOnpp
MIME Tools
NppExport
NppFTP
Plugin Manager
Show NppFTP Window
Focus NppFTP Window
About NppFTP
Normal text file
length : 0 lines : 1
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Search Windows
2:38 PM
8/17/2017


new 1 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
new 1
NppFTP - Disconnected
Profile settings
Profiles:
math209
Connection
Authentication
Transfers
FTP Misc.
Cache
Hostname:
math209.cs.uno.edu
Connection type:
SFTP

How to use Notepad++ or Sublime Text for our labs

To use these editors on your own computer: see resources section of class website for info.
`http://2467.cs.uno.edu/resources.html`

Notepad++, Sublime, and Atom have ways to use SFTP to synchronize the edits you make with your files on the `systems-lab` server.

Atom

student.c — ~/2467/introlab/part1 — Atom

File Edit View Selection Find Packages Help

Telemetry Consent | Welcome | Welcome Guide | student.c

ATOM

A hackable text editor for the 21st Century

For help, please visit

- The Atom docs for Guides and the API reference.
- The Atom forum at discuss.atom.io
- The Atom org. This is where all GitHub-created Atom packages can be found.

☑ Show Welcome Guide when opening Atom

atom.io ×

```c
#include <string.h>
#include "student.h"

struct student_info make_student(void){
   struct student_info me;

   me.id = 2234;
   strcpy(me.name, "My Name is not My Name"); /* strcpy is necessary */

   me.csci_classes[0]=2120; /* array indices always being with 0 */
   me.csci_classes[1]=2451;

   strcpy(me.reason, "All the cool kids are doing it.");

   return me;
}
```

~/2467/introlab/part1/student.c* 13:54

VS Code

student.c - Visual Studio Code

File Edit Selection View Go Debug Terminal Help

student.c

home > matoups1 > 2467 > introlab > part1 > student.c

```c
#include <string.h>
#include "student.h"

struct student_info make_student(void){
    struct student_info me;

    me.id = 2234;
    strcpy(me.name, "My Name is not My Name"); /* strcpy is necessary */

    me.csci_classes[0]=2120; /* array indices always being with 0 */
    me.csci_classes[1]=2451;

    strcpy(me.reason, "All the cool kids are doing it.");

    return me;
}
```

The 'C/C++' extension is recommended for this file type.

Install Show Recommendations

0 0 Ln 1, Col 1 Spaces: 3 UTF-8 LF C 1

Bottom line on editors

- You will be using a text editor often in this course, for both programming and many other tasks.
- You should have a least some familiarity with a text-based editor such as `nano` or `vim`
  - `nano` is better to start with, `vim` will take much more time to learn.
- You may also benefit from learning a graphical editor such as the others we mentioned. (Optional)
  - One benefit: less latency due to network connection to `systems-lab` when working from elsewhere.
- This skill will be important for you beyond this course.
  - This is a good opportunity to build your skills, but this course is not about Text Editors. After this week we will not discuss this further during class, and we will expect you to be able to edit files on your own.
- Ask questions now! Your course staff and your fellow students are all good sources for tips and tricks on editors.

Overview

2. Lab 0
   - Prerequisites review
   - Logging in to computer lab terminals
   - Remote access with `ssh`
   - The terminal
   - Text editors
   - Part 1
   - Part 2
   - Part 3
   - Handin notes

Part 1: Who are you?

```
  GNU nano 2.3.1              File: student.h

/* student.h */
/* define a struct for storing student information */

struct student_info {
   int id;
   char name[80]; /* 80 characters should be more than enough to store a name, $

   /* need more stuff here */
   int csci_classes[20]; /* has anyone taken more than 20 CS classes? */

   char reason[400]; /* This means my reason for studying CS must be
                        less than 400 characters, or I will have to
                        increase the array size. */
};

/* this is a function prototype, the function itself is in student.c */
struct student_info make_student(void);



^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Part 1: Who are you?

```
  GNU nano 2.3.1              File: student.c

#include <string.h>
#include "student.h"

struct student_info make_student(void){
   struct student_info me;

   me.id = 2222222;
   strcpy(me.name, "My Name"); /* strcpy is necessary */

   me.csci_classes[0]=2120; /* array indices always being with 0 */
   me.csci_classes[1]=2450;

   strcpy(me.reason, "I meant to major in Counselor Education, but I clicked on$

   return me;
}



                          [ Read 16 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Part 1: Who are you?

```
  GNU nano 2.5.3                    File: part1.c

/* part1.c -- This is part 1 of lab 0, CSCI2467, Fall 2016 */

#include <stdio.h>
#include "student.h"

int main () /* in this case we don't care about command line arguments */
{
        /* create a struct of type student_info called student */
        /* then call the make_student() function from student.c
         * to fill it with information */
  struct student_info student = make_student ();

  /* Now start printing out what's stored in student.
   * \n represents the newline character
   * %d is for substituting integer values */
  printf ("ID number: %d\n\n", student.id);

  /* %s is for substituting a character array */
  printf ("Name: %s\n\n", student.name);

  /* we reference individual array elements with [0] [1] etc */
  printf ("CS Courses taken: CSCI%d, CSCI%d\n\n",
          student.csci_classes[0], student.csci_classes[1]);

  printf ("Reason for studying Computer Science:\n%s\n\n", student.reason);

  return 0; /* 0 is typically returned when no errors have occurred */
};

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line
```

Part 1: Compiling and running

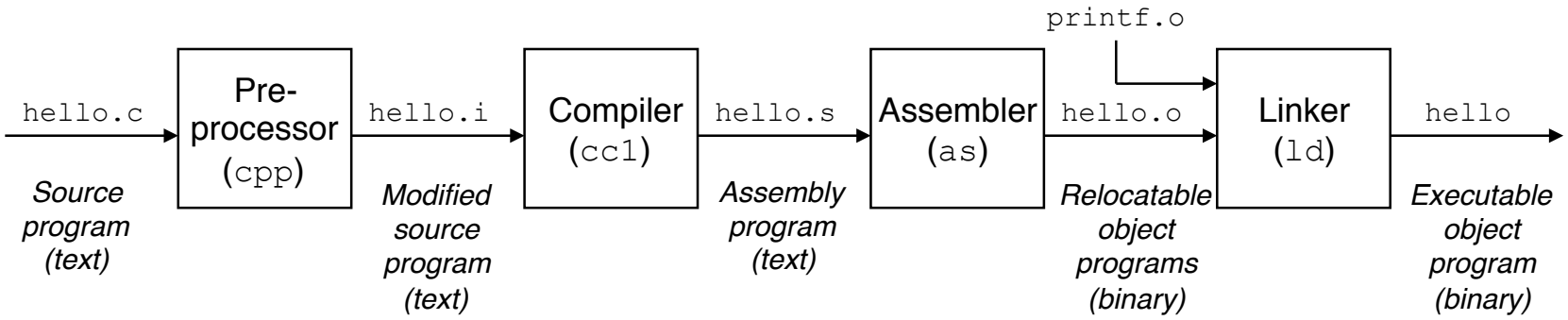
- Save (in text editor)
- Compile with `gcc` as given in lab manual
- Did compilation succeed? (No errors means yes)
- Run with `./part1`

Part 2: lifecycle of a C program

A slightly modified `hello.c`

```
  GNU nano 2.2.6            File: hello.c

#include <stdio.h>
#define RETURNVALUE 67

int main()
{
    int retval = RETURNVALUE;

    printf("Hello world.\n");

    return retval;
}

^G Get Help^O WriteOut^R Read Fil^Y Prev Pag^K Cut Text^C Cur Pos
^X Exit    ^J Justify ^W Where Is^V Next Pag^U UnCut Te^T To Spell
```

Part 2: lifecycle of a C program



Source:

Bryant & O'Hallaron (2003)

Part 3: How long does it take to copy an array?

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (i = 0; i < 2048; i++)
    for (j = 0; j < 2048; j++)
      dst[i][j] = src[i][j];
}
```

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

Source:

Bryant & O'Hallaron (2003)

Part 3: How long does it take to copy an array?

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (i = 0; i < 2048; i++)
    for (j = 0; j < 2048; j++)
      dst[i][j] = src[i][j];
}
```

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

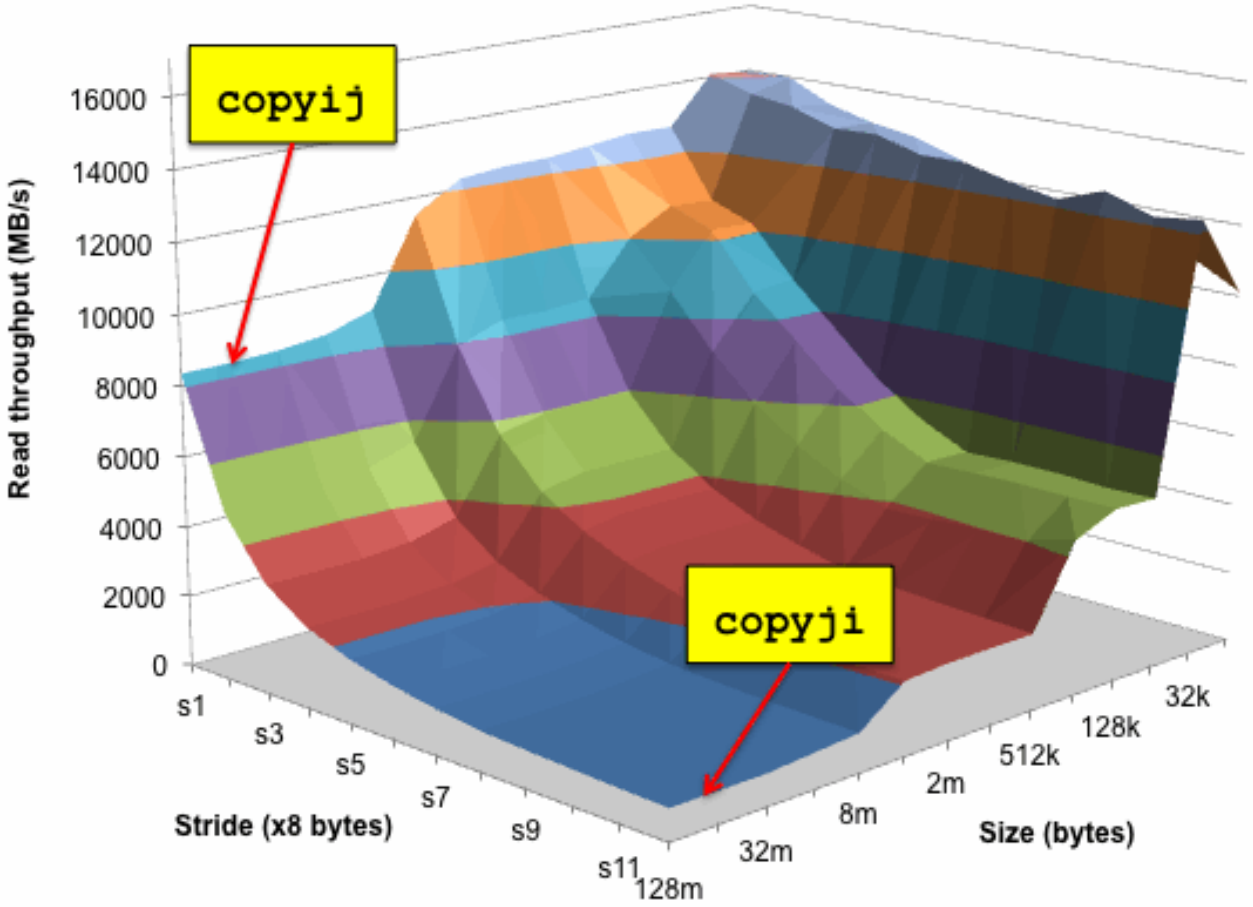
4.3ms **81.8ms**

2.0 GHz Intel Core i7 Haswell

Source:

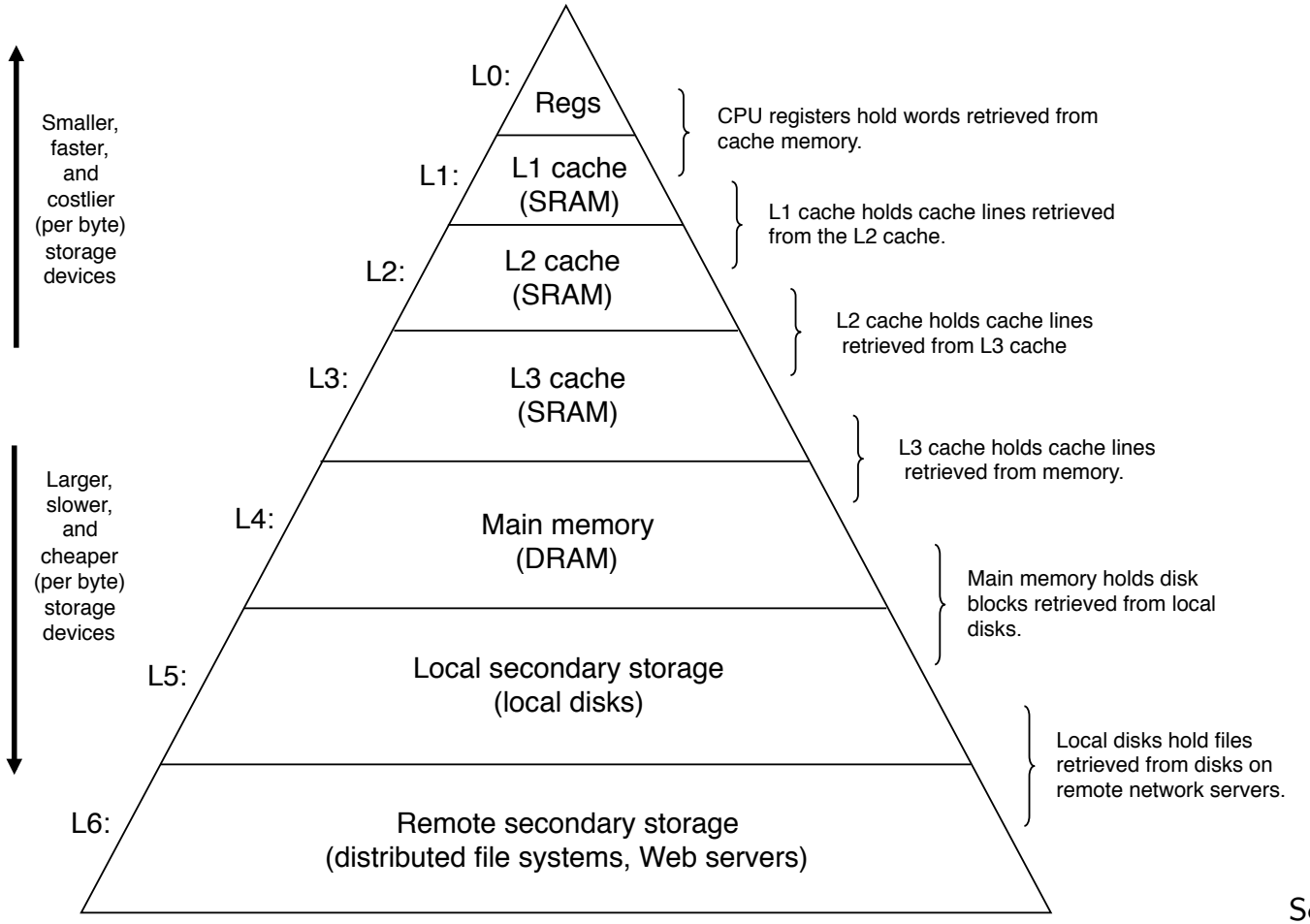
Bryant & O'Hallaron (2003)

Part 3: "Memory mountain"



Source:

Part 3: Memory hierarchy from Chapter 1



Source: Bryant & O'Hallaron (2003)

Wrapping up: Handing in your work

- We will use Autolab for handin
  - use a `.tar` file, not `.zip`
  - specific details given in introlab writeup
- Instant feedback from Autolab
  - you'll know right away if there is something obviously missing from your submission
  - course staff still reads and grades your assignment, so Autolab is not the final word on your score
  - don't try to game the system by putting in fake answers or comments, we *will* notice

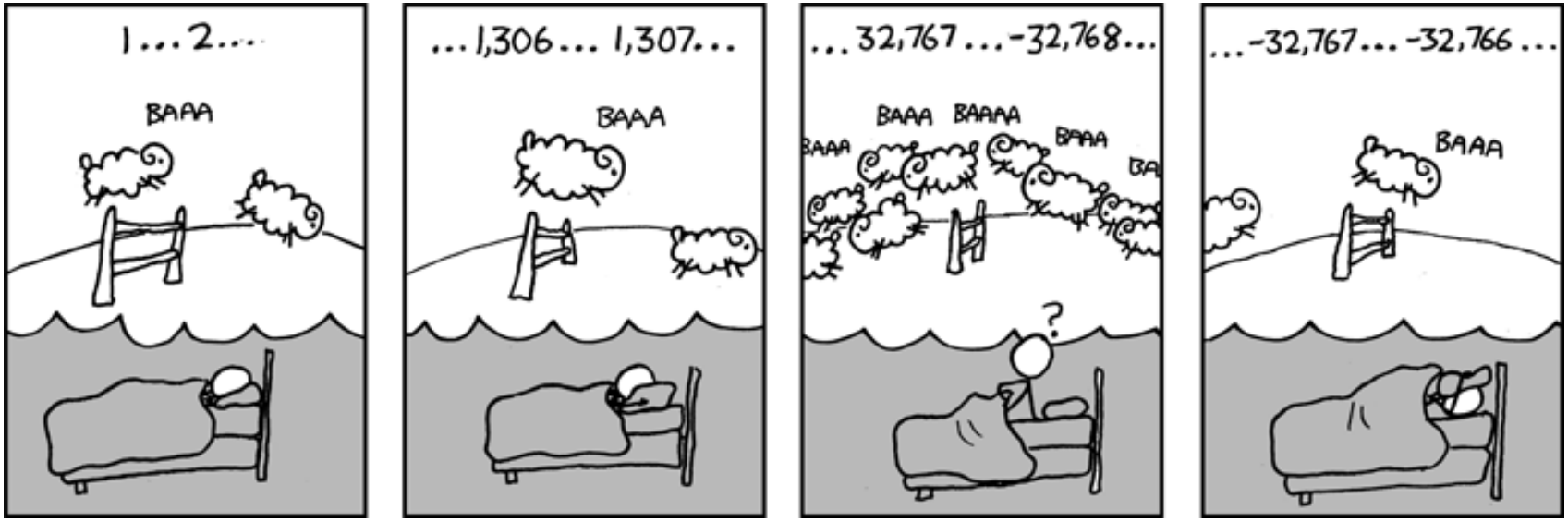
4. Preview of next class
   - Reading
   - Data surprises!

Chapter 2 is long, start today!

Get started on readings for next classes:

- Read section 2.1 for Wednesday Jan 22
- Section 2.2 for Friday Jan 24

ints are not Integers



Source: xkcd.com

$\mathbb{Z}$ is infinitely large, computer memory is not.

This is the fundamental challenge!