```
$ cat welcome.c
#include <stdio.h>

int main(int argc, char *argv[])
{
  printf(“COS 217\n”);
  printf(“Introduction to Programming Systems\n\n”);

  printf(“Spring, 2018\n”);
  return 0;
}

$ gcc217 welcome.c –o welcome

$ ./welcome
COS 217
Introduction to Programming Systems


Spring, 2018
```

Agenda

Course overview

- **Introductions**
- Course goals
- Resources
- Grading
- Policies
- Schedule

Getting started with C

- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)

Introductions

Lead Instructor

- Prof. Szymon Rusinkiewicz smr@princeton.edu

Lead Preceptor

- Robert Dondero, Ph.D. rdondero@cs.princeton.edu

Faculty Preceptor

- Donna Gabai dgabai@cs.princeton.edu

Preceptors

- Seo Young Kyung skyung@princeton.edu
- Austin Le austinle@princeton.edu

Agenda

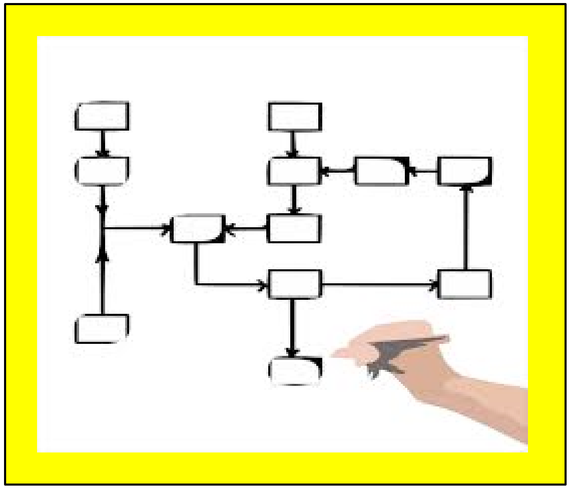
Course overview

- Introductions
- **Course goals**
- Resources
- Grading
- Policies
- Schedule

Getting started with C

- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)

Goal 1: Programming in the Large



Goal 1: "Programming in the large"

- Help you learn how to compose large computer programs

Topics

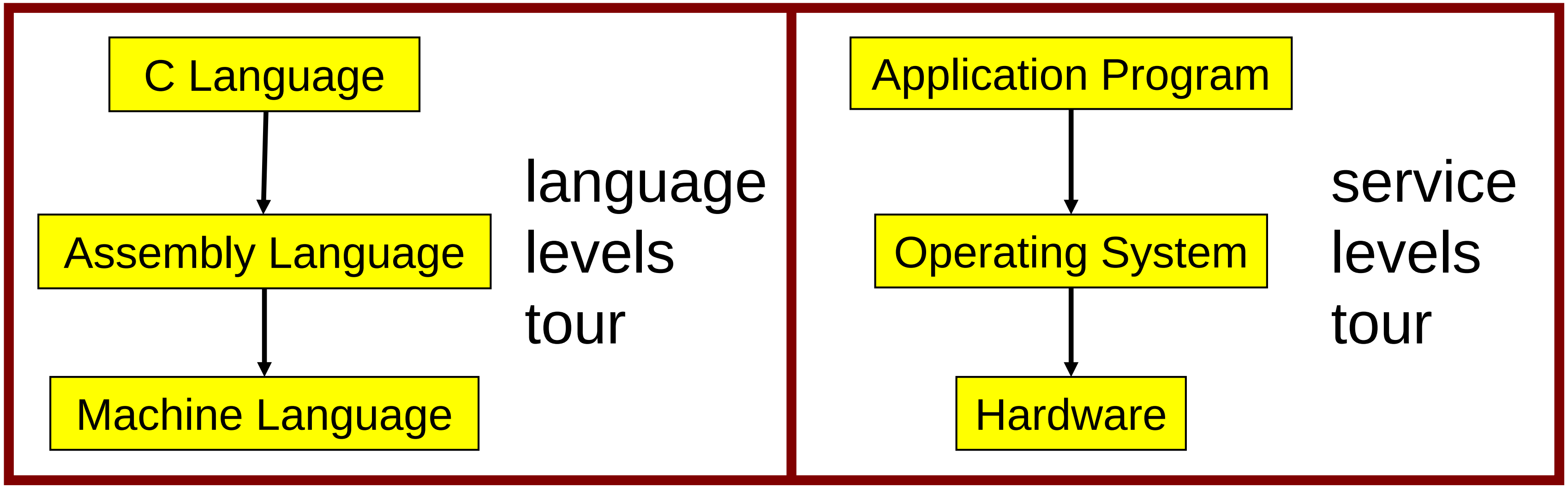
- Modularity/abstraction, information hiding, resource management, error handling, testing, debugging, performance improvement, tool support

Goal 2: Under the Hood

Learn what happens "under the hood" of computer systems

Learn "how to be a client of an operating system"

Downward tours



C Language

Assembly Language

Machine Language

language levels tour

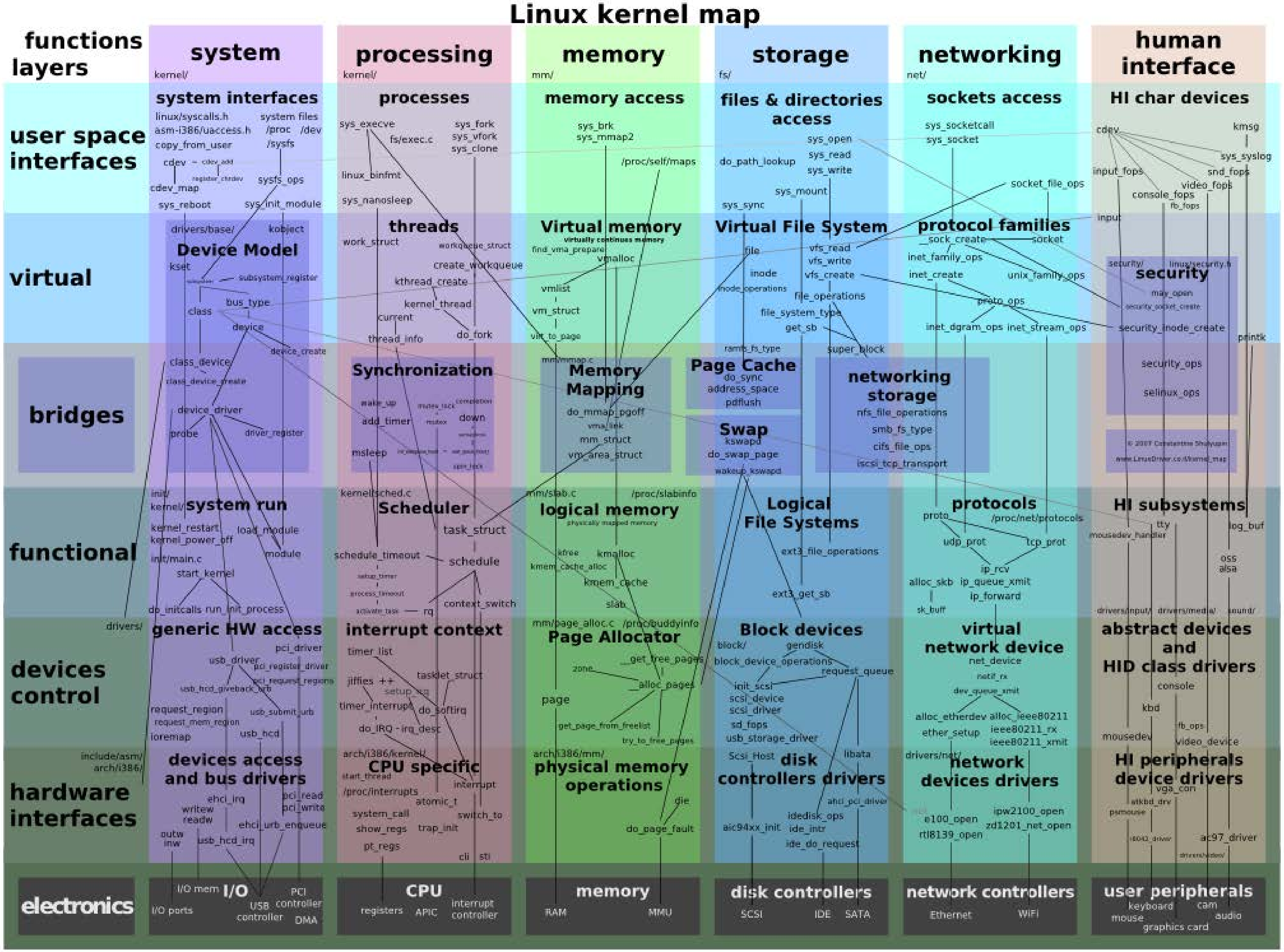
Application Program

Operating System

Hardware

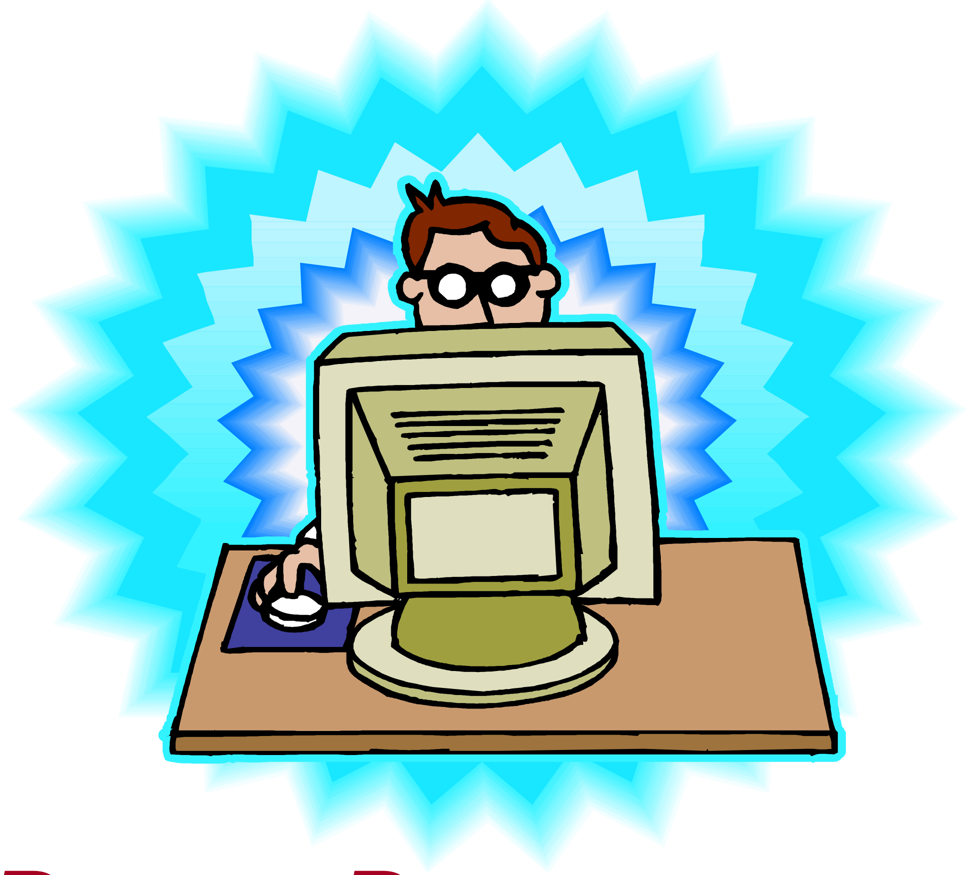
service levels tour

Modularity!


Linux kernel map
functions layers
system
kernel/
processing
kernel/
memory
mm/
storage
fs/
networking
net/
human interface
user space interfaces
system interfaces
linux/syscalls.h
asm-i386/uaccess.h
copy_from_user
system files
/proc /dev
/sysfs
cdev
cdev_add
register_chrdev
cdev_map
sysfs_ops
sys_reboot
sys_init_module
processes
sys_execve
fs/exec.c
sys_fork
sys_vfork
sys_clone
linux_binfmt
sys_nanosleep
memory access
sys_brk
sys_mmap2
/proc/self/maps
files & directories access
sys_open
sys_read
sys_write
do_path_lookup
sys_mount
sys_sync
sockets access
sys_socketcall
sys_socket
socket_file_ops
HI char devices
cdev
kmsg
sys_syslog
input_fops
snd_fops
video_fops
console_fops
fb_fops
input
virtual
drivers/base/
kobject
Device Model
kset
subsystem_register
bus_type
class
device
device_create
threads
work_struct
workqueue_struct
create_workqueue
kthread_create
kernel_thread
current
thread_info
do_fork
Virtual memory
virtually continues memory
find_vma_prepare
vmalloc
vmlist
vm_struct
virt_to_page
Virtual File System
file
vfs_read
vfs_write
vfs_create
inode
inode_operations
file_operations
file_system_type
get_sb
protocol families
sock_create
socket
inet_family_ops
inet_create
unix_family_ops
proto_ops
inet_dgram_ops
inet_stream_ops
security/
linux/security.h
security
may_open
security_socket_create
security_inode_create
printk
bridges
class_device
class_device_create
device_driver
probe
driver_register
Synchronization
wake_up
mutex_lock
completion
add_timer
mutex
down
semaphore
msleep
spin_lock
mm/mmap.c
Memory Mapping
do_mmap_pgoff
vma_link
mm_struct
vm_area_struct
ramfs_fs_type
super_block
Page Cache
do_sync
address_space
pdflush
Swap
kswapd
do_swap_page
wakeup_kswapd
networking storage
nfs_file_operations
smb_fs_type
cifs_file_ops
iscsi_tcp_transport
security_ops
selinux_ops
© 2007 Constantine Shulyupin
www.LinuxDriver.co.il/kernel_map
functional
init/
kernel/
system run
kernel_restart
kernel_power_off
load_module
module
init/main.c
start_kernel
do_initcalls
run_init_process
kernel/sched.c
Scheduler
task_struct
schedule_timeout
schedule
setup_timer
process_timeout
activate_task
rq
context_switch
mm/slab.c
/proc/slabinfo
logical memory
physically mapped memory
kfree
kmalloc
kmem_cache_alloc
kmem_cache
slab
Logical File Systems
ext3_file_operations
ext3_get_sb
protocols
proto
/proc/net/protocols
udp_prot
tcp_prot
ip_rcv
alloc_skb
ip_queue_xmit
ip_forward
sk_buff
HI subsystems
tty
log_buf
mousedev_handler
oss
alsa
devices control
drivers/
generic HW access
pci_driver
usb_driver
pci_register_driver
pci_request_regions
usb_hcd_giveback_urb
request_region
usb_submit_urb
request_mem_region
usb_hcd
ioremap
interrupt context
timer_list
jiffies ++
tasklet_struct
setup_irq
timer_interrupt
do_softirq
do_IRQ
irq_desc
mm/page_alloc.c
/proc/buddyinfo
Page Allocator
zone
get_free_pages
alloc_pages
page
get_page_from_freelist
try_to_free_pages
Block devices
block/
gendisk
block_device_operations
request_queue
init_scsi
scsi_device
scsi_driver
sd_fops
usb_storage_driver
virtual network device
net_device
netif_rx
dev_queue_xmit
alloc_etherdev
alloc_ieee80211
ether_setup
ieee80211_rx
ieee80211_xmit
drivers/input/
drivers/media/
sound/
abstract devices and HID class drivers
console
kbd
fb_ops
mousedev
video_device
hardware interfaces
include/asm/
arch/i386/
devices access and bus drivers
ehci_irq
writew
readw
pci_read
pci_write
ehci_urb_enqueue
outw
inw
usb_hcd_irq
arch/i386/kernel/
CPU specific
start_thread
/proc/interrupts
interrupt
atomic_t
system_call
switch_to
show_regs
trap_init
pt_regs
cli
sti
arch/i386/mm/
physical memory operations
die
do_page_fault
Scsi_Host
disk controllers drivers
libata
ahci_pci_driver
aic94xx_init
idedisk_ops
ide_intr
ide_do_request
drivers/net/
network devices drivers
e100_open
ipw2100_open
zd1201_net_open
rtl8139_open
HI peripherals device drivers
vga_con
atkbd_drv
psmouse
i8042_driver
ac97_driver
drivers/video/
electronics
I/O mem
I/O
PCI
USB controller
PCI controller
I/O ports
DMA
CPU
registers
APIC
interrupt controller
memory
RAM
MMU
disk controllers
SCSI
IDE
SATA
network controllers
Ethernet
WiFi
user peripherals
keyboard
cam
mouse
audio
graphics card

Goals: Summary

Help you to become a...



Power Programmer!!!

Goals: Why C?

Question: Why C instead of Java?

Answer 1: Primary language for “under the hood” programming

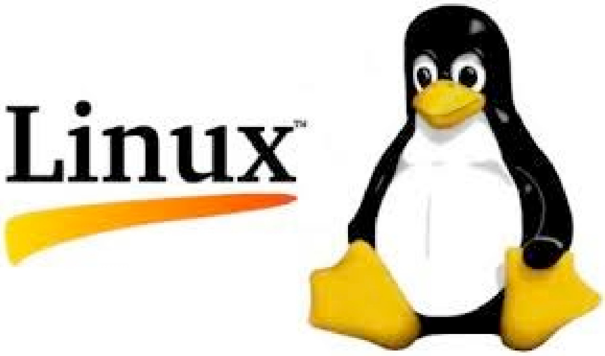
Answer 2: Knowing a variety of approaches helps you “program in the large”

Goals: Why Linux?

Question: Why use the Linux operating system?

Answer 1: Linux is good for education and research

Answer 2: Linux (with GNU tools) is good for programming



Agenda

Course overview

- Introductions
- Course goals
- **Resources**
- Grading
- Policies
- Schedule

Getting started with C

- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)

Lectures

Lectures

- Describe material at conceptual (high) level
- Slides available via course website

Lecture etiquette

- Use electronic devices *only* for taking notes or annotating slides
- No FaceNewsChatBookSnapMail, please

iClicker

- Please obtain one and register in Blackboard
  (not with iClicker – they'll charge you)
- Occasional questions in class, graded on participation
  (with a generous allowance for not being able to attend)

iClicker Question

Q: Do you have an iClicker with you today?

- A. Yes
- B. No, but I’ve been practicing my mental electrotelekinesis and the response is being registered anyway
- C. I’m not here, but someone is iClicking for me (don’t do this!)

Precepts

Precepts

- Describe material at the “practical” (low) level
- Support your work on assignments
- Hard copy handouts distributed during precepts
- Handouts available via course website

Precept etiquette

- Attend your precept – attendance will be taken
- Use SCORE to move to another precept
  - Trouble ⇒ See Colleen Kenny (CS Bldg 210)
    - But Colleen can’t move you into a full precept
- Must miss your precept? ⇒ inform preceptors & attend another

Precepts begin today and tomorrow!

Website

Website

- Access from http://www.cs.princeton.edu/
  - Princeton CS → Courses → Course Schedule → COS 217
  - Home page, schedule page, assignment page, policies page

Piazza

Piazza

- http://piazza.com/class#spr2018/cos217/
- Instructions provided in first precept

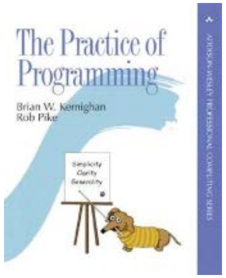
Piazza etiquette

- Study provided material before posting question
  - Lecture slides, precept handouts, required readings
- Read all (recent) Piazza threads before posting question
- Don’t show your code!!!
  - See course policies

Books

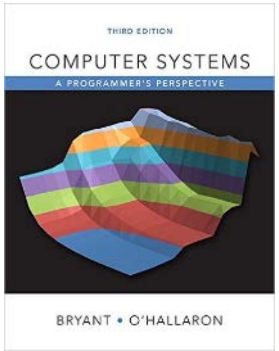
The Practice of Programming (recommended)

- Kernighan & Pike
- “Programming in the large”



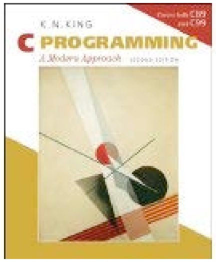
Computer Systems: A Programmer’s Perspective (Third Edition) (recommended)

- Bryant & O'Hallaron
- “Under the hood”



C Programming: A Modern Approach (Second Edition) (required)

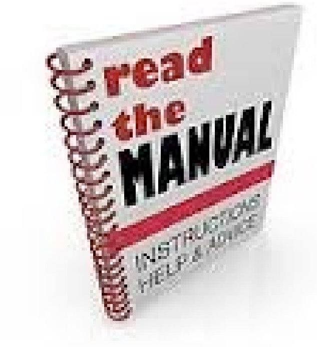
- King
- C programming language and standard libraries



Manuals

Manuals (for reference only, available online)

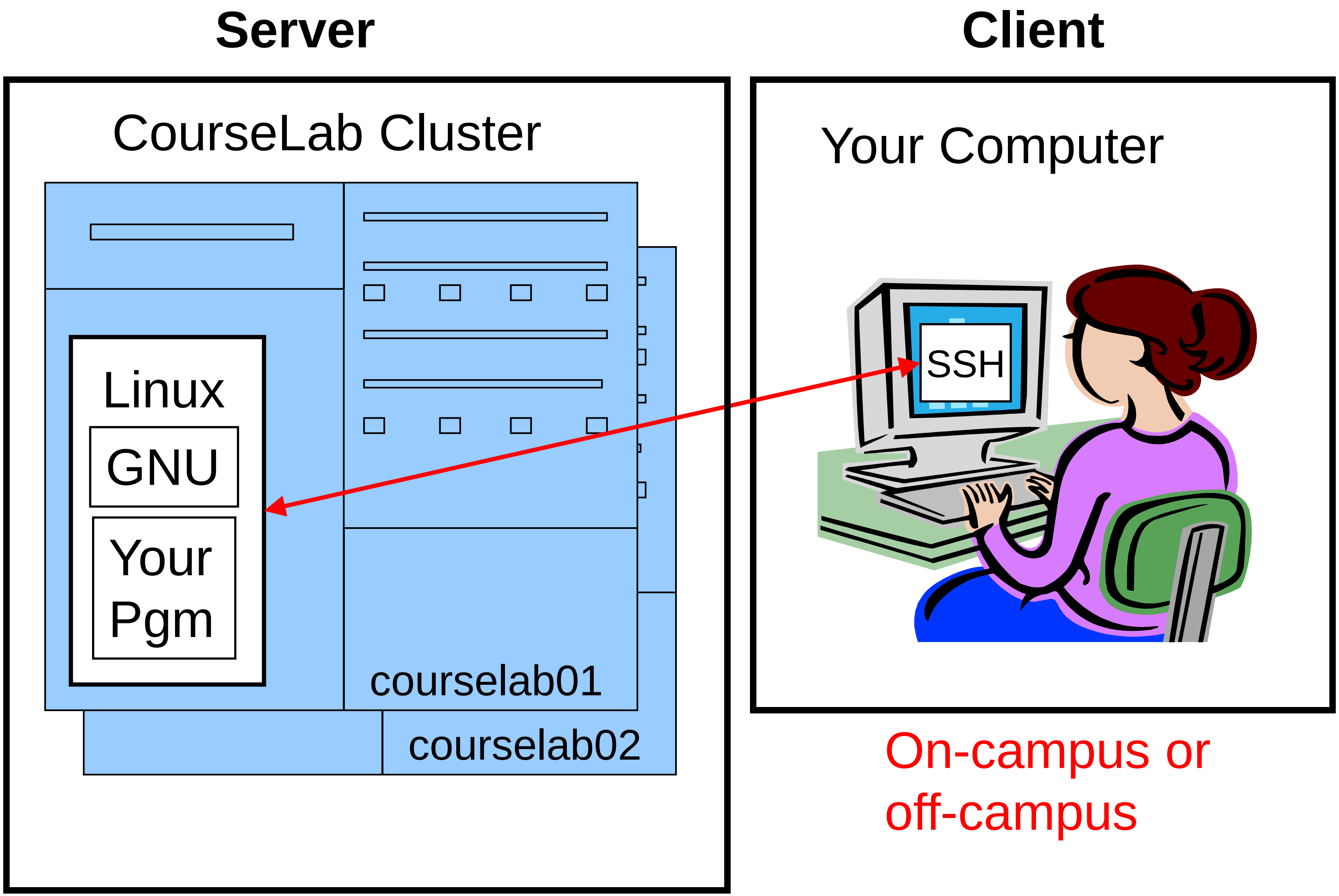
- *Intel 64 and IA-32 Architectures Software Developer's Manual, Volumes 1-3*
- *Intel 64 and IA-32 Architectures Optimization Reference Manual*
- *Using* `as`*, the GNU Assembler*



See also

- Linux `man` command

Programming Environment



On-campus or off-campus

Agenda

Course overview

- Introductions
- Course goals
- Resources
- **Grading**
- Policies
- Schedule

Getting started with C

- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)

Grading

Course Component	Percentage of Grade
Assignments *	50
Midterm Exam **	15
Final Exam **	25
Participation ***	10

These percentages are approximate

* Final assignment counts double; penalties for lateness

** Closed book, closed notes, no electronic devices

*** Did your involvement benefit the course as a whole?

- Lecture/precept attendance and participation counts

Programming Assignments

Programming assignments

(some individual, some done with a partner from your precept)

0. Introductory survey
1. “De-comment” program
2. String module
3. Symbol table module
4. Assembly language programs
5. Buffer overrun attack
6. Heap manager module
7. Unix shell

Assignments 0 and 1 are available now

Start early!!!

Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- **Policies**
- Schedule

Getting started with C

- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)

Policies

Study the course "Policies" web page!

Especially the assignment collaboration policies

- Violations often involve **trial by Committee on Discipline**
- Typical course-level penalty is **F for course**
- Typical University-level penalty is **suspension from University** for 1 academic year

Assignment Related Policies

Some highlights:

- You may not reveal any of your assignment solutions (products, descriptions of products, design decisions) on Piazza.
- **Getting help**: To help you compose an assignment solution you may use only authorized sources of information, may consult with other people only via the course's Piazza account or via interactions that might legitimately appear on the course's Piazza account, and must declare your sources in your readme file for the assignment.
- **Giving help**: You may help other students with assignments only via the course's Piazza account or interactions that might legitimately appear on the course's Piazza account, and you may not share your assignment solutions with anyone, ever, in any form.

Ask the instructor for clarifications

- Permission to deviate from policies must be obtained in writing

Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies
- **Schedule**

Getting started with C

- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)

Course Schedule

Weeks	Lectures	Precepts
1-2	Number Systems C (conceptual)	Linux/GNU C (pragmatic)
3-6	Programming in the Large	Advanced C
6	Midterm Exam	
7	Spring break!	
8-13	“Under the Hood” (conceptual)	“Under the Hood” (assignment how-to)
	Reading Period	
	Final Exam	

Questions?

Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies
- Schedule

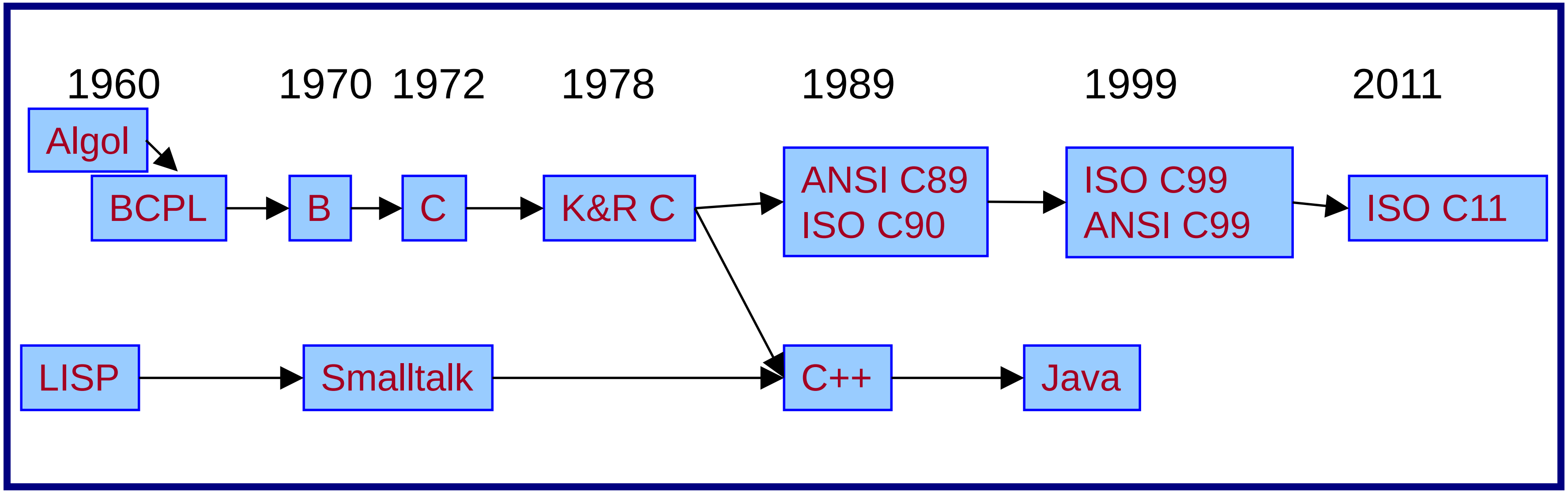
Getting started with C

- **History of C**
- Building and running C programs
- Characteristics of C
- C details (if time)

The C Programming Language

Who?	Dennis Ritchie
When?	~1972
Where?	Bell Labs
Why?	Build the Unix OS

Java vs. C: History



C vs. Java: Design Goals

C Design Goals (1975)	Java Design Goals (1995)
Build the Unix OS	Language of the Internet
Low-level; close to HW and OS	High-level; insulated from hardware and OS
Good for system-level programming	Good for application-level programming
Support structured programming	Support object-oriented programming
Unsafe: don’t get in the programmer’s way	Safe: can’t step “outside the sandbox”
	Look like C!

Agenda

Course overview

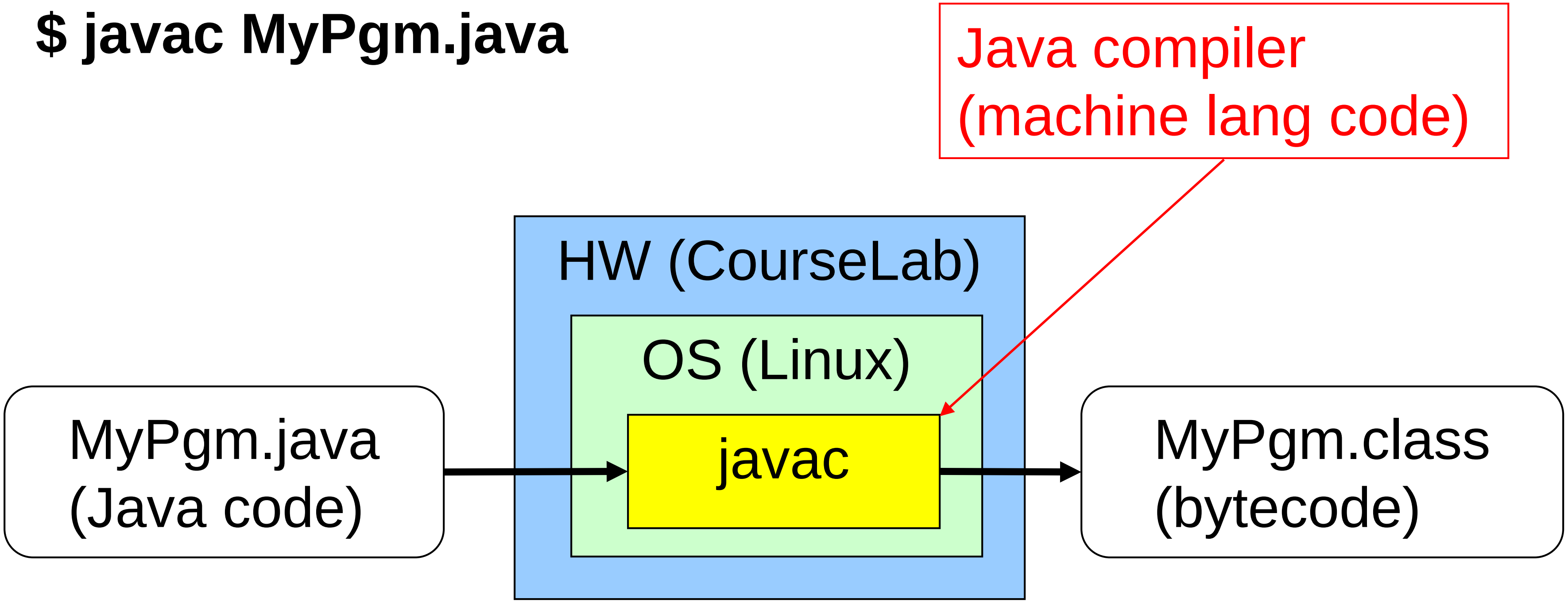
- Introductions
- Course goals
- Resources
- Grading
- Policies
- Schedule

Getting started with C

- History of C
- **Building and running C programs**
- Characteristics of C
- C details (if time)

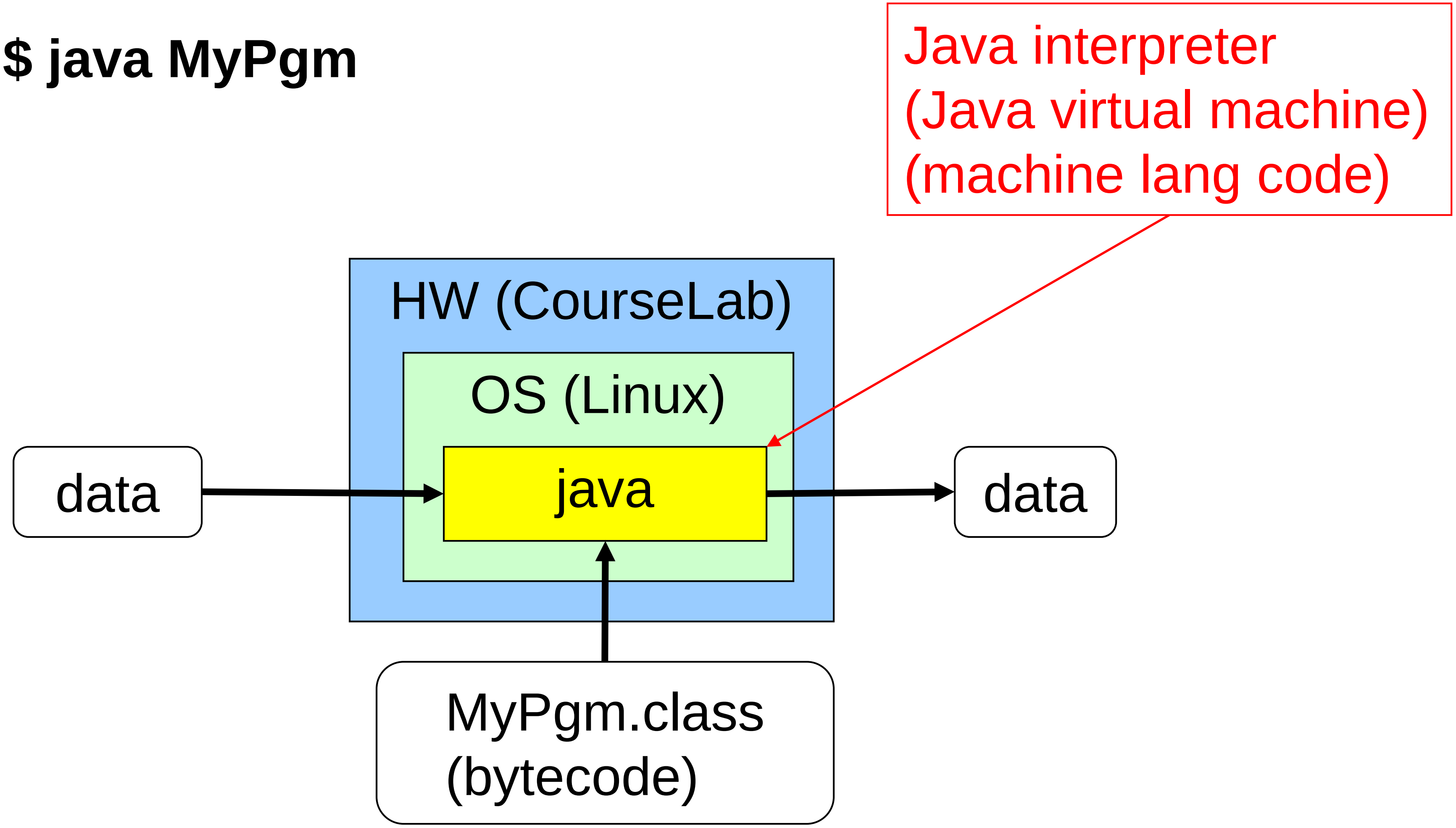
Building Java Programs

```
$ javac MyPgm.java
```



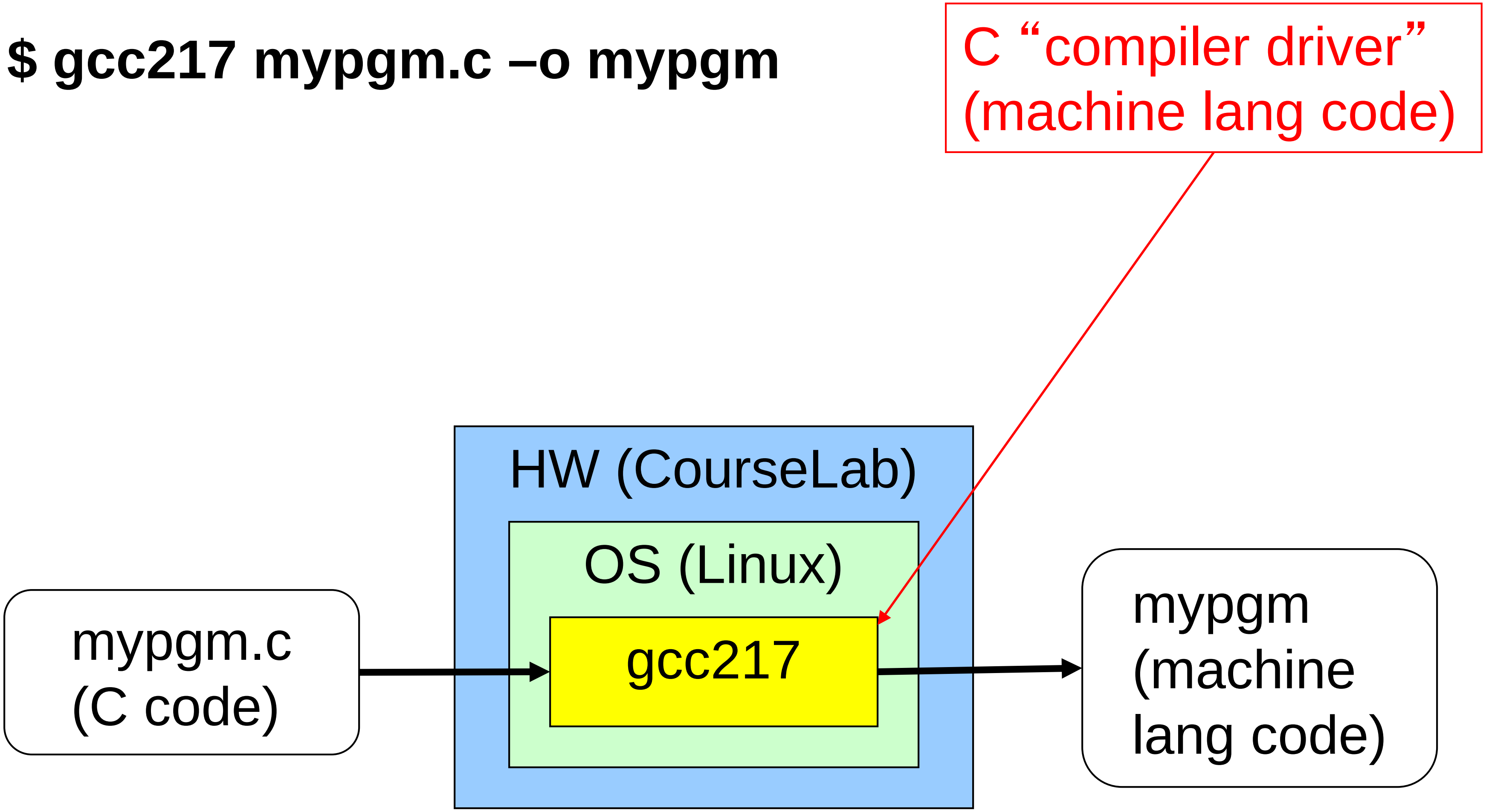
Running Java Programs

$ java MyPgm



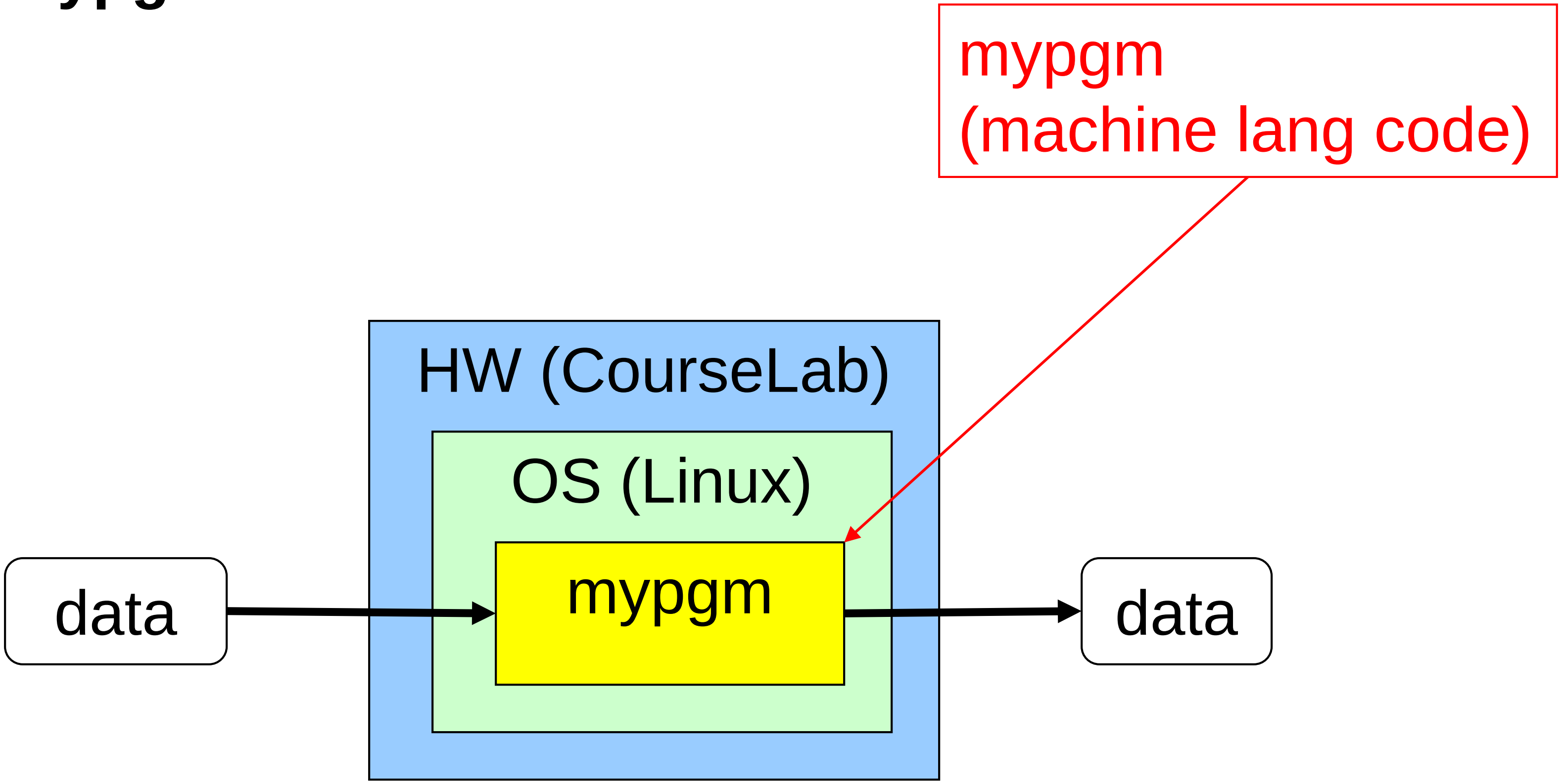
Building C Programs

$ gcc217 mypgm.c –o mypgm



Running C Programs

$ ./mypgm



Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies
- Schedule

Getting started with C

- History of C
- Building and running C programs
- **Characteristics of C**
- C details (if time)

Java vs. C: Portability

Program	Code Type	Portable?
MyPgm.java	Java source code	Yes
mypgm.c	C source code	Mostly
MyPgm.class	Bytecode	Yes
mypgm	Machine lang code	No

Conclusion: Java programs are more portable

Java vs. C: Safety & Efficiency

Java

- Automatic array-bounds checking,
- NULL pointer checking,
- Automatic memory management (garbage collection)
- Other safety features

C

- Manual bounds checking
- NULL pointer checking,
- Manual memory management

Conclusion 1: Java is often safer than C

Conclusion 2: Java is often slower than C

Java vs. C: Characteristics

	Java	C
Portability	+	-
Efficiency	~	+
Safety	+	-

iClicker Question

Q: Which corresponds to the C programming language?

- A.

- B.

- C.

Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies
- Schedule

Getting started with C

- History of C
- Building and running C programs
- Characteristics of C
- **C details (if time)**

Java vs. C: Details

Remaining slides provide some details

Use for future reference

Slides covered now, as time allows…

Java vs. C: Details

	Java	C

Overall Program Structure	`Hello.java:`
	
	`public class Hello`
	`{  public static void main`
	`         (String[] args)`
	`   {  System.out.println(`
	`            "hello, world");`
	`   }`
	`}`
		`hello.c:`
		
		`#include <stdio.h>`
		
		`int main(void)`
		`{  printf("hello, world\n");`
		`   return 0;`
		`}`

Building	`$ javac Hello.java`	`$ gcc217 hello.c –o hello`

Running	`$ java Hello`
	`hello, world`
	`$`
		`$ ./hello`
		`hello, world`
		`$`

Java vs. C: Details

	Java	C
Character type	char // 16-bit Unicode	char /* 8 bits */
Integral types	byte // 8 bits short // 16 bits int // 32 bits long // 64 bits	(unsigned) char (unsigned) short (unsigned) int (unsigned) long
Floating point types	float // 32 bits double // 64 bits	float double long double
Logical type	boolean	/* no equivalent */ /* use integral type */
Generic pointer type	Object	void*

Constants	final int MAX = 1000;	#define MAX 1000
		const int MAX = 1000;
		enum {MAX = 1000};

Java vs. C: Details

	Java	C

Arrays	`int [] a = new int [10];`
	`float [][] b =`
	`   new float [5][20];`
		`int a[10];`
		`float b[5][20];`

Array bound checking	`// run-time check`	`/* no run-time check */`

Pointer type	`// Object reference is an`
	`// implicit pointer`
		`int *p;`

Record type	`class Mine`
	`{  int x;`
	`   float y;`
	`}`
		`struct Mine`
		`{  int x;`
		`   float y;`
		`};`

Java vs. C: Details

<table>
<tr><th></th><th>Java</th><th>C</th></tr>
<tr><td>Strings</td><td>String s1 = "Hello";
String s2 = new
String("hello");</td><td>char *s1 = "Hello";
char s2[6];
strcpy(s2, "hello");</td></tr>
<tr><td>String concatenation</td><td>s1 + s2
s1 += s2</td><td>#include <string.h>
strcat(s1, s2);</td></tr>
<tr><td>Logical ops *</td><td>&amp;&amp;, ||, !</td><td>&amp;&amp;, ||, !</td></tr>
<tr><td>Relational ops *</td><td>=, !=, &gt;, <, &gt;=, &lt;=</td><td>=, !=, &gt;, <, &gt;=, &lt;=</td></tr>
<tr><td>Arithmetic ops *</td><td>+, -, *, /, %, unary -</td><td>+, -, *, /, %, unary -</td></tr>
<tr><td>Bitwise ops</td><td>&gt;&gt;, &lt;&lt;, &gt;&gt;&gt;, &amp;, |, ^</td><td>&gt;&gt;, &lt;&lt;, &amp;, |, ^</td></tr>
<tr><td>Assignment ops</td><td>=, *=, /=, +=, -=, <<=,
>>=, >>>=, =, &amp;=, ^=, |=,
%=</td><td>=, *=, /=, +=, -=, <<=,
>>=, =, &amp;=, ^=, |=, %=</td></tr>
</table>

* Essentially the same in the two languages

Java vs. C: Details

	Java	C

if stmt *	`if (i < 0)`
	`   statement1;`
	`else`
	`   statement2;`
		`if (i < 0)`
		`   statement1;`
		`else`
		`   statement2;`

switch stmt *	`switch (i)`
	`{  case 1:`
	`      ...`
	`      break;`
	`   case 2:`
	`      ...`
	`      break;`
	`   default:`
	`      ...`
	`}`
		`switch (i)`
		`{  case 1:`
		`      ...`
		`      break;`
		`   case 2:`
		`      ...`
		`      break;`
		`   default:`
		`      ...`
		`}`

goto stmt	`// no equivalent`	`goto someLabel;`

* Essentially the same in the two languages

Java vs. C: Details

	Java	C

for stmt	for (int i=0; i<10; i++)
	    statement;
		int i;
		for (i=0; i<10; i++)
		    statement;

while stmt *	while (i < 0)
	    statement;
		while (i < 0)
		    statement;

do-while stmt *	do
	    statement;
	while (i < 0)
		do
		    statement;
		while (i < 0);

continue stmt *	continue;	continue;
labeled continue stmt	continue *someLabel*;	/* no equivalent */
break stmt *	break;	break;
labeled break stmt	break *someLabel*;	/* no equivalent */

* Essentially the same in the two languages

Java vs. C: Details

	Java	C

return stmt *	return 5;
	return;
		return 5;
		return;

Compound stmt (alias block) *	{
	statement1;
	statement2;
	}
		{
		statement1;
		statement2;
		}

Exceptions	throw, try-catch-finally	/* no equivalent */
Comments	/* comment */ // another kind	/* comment */

Method / function call	f(x, y, z);
	someObject.f(x, y, z);
	SomeClass.f(x, y, z);
		f(x, y, z);

* Essentially the same in the two languages

Example C Program

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{  const double KMETERS_PER_MILE = 1.609;
   int miles;
   double kMeters;

   printf("miles: ");
   if (scanf("%d", &miles) != 1)
   {  fprintf(stderr, "Error: Expected a number.\n");
      exit(EXIT_FAILURE);
   }

   kMeters = (double)miles * KMETERS_PER_MILE;
   printf("%d miles is %f kilometers.\n",
      miles, kMeters);
   return 0;
}
```

Summary

Course overview

- Introductions
- Course goals
  - Goal 1: Learn “programming in the large”
  - Goal 2: Look “under the hood”and learn low-level programming
  - Use of C and Linux supports both goals
- Resources
  - Lectures, precepts, programming environment, Piazza, textbooks
  - Course website: access via http://www.cs.princeton.edu
- Grading
- Policies
- Schedule

Summary

Getting started with C

- History of C
- Building and running C programs
- Characteristics of C
- Details of C
  - Java and C are similar
  - Knowing Java gives you a head start at learning C

Getting Started

Check out course website **soon**

- **Study “Policies” page**
- First assignment is available

Establish a reasonable computing environment **soon**

- Instructions given in first precept