UNIVERSITY *of* WEST FLORIDA

COP4634: Systems & Networks I

Course Overview

C++, Unix, Compiling, g++, …

Thomas Reichherzer

treichherzer@uwf.edu

04/227   (850) 474-2612

See syllabus for office hours

- Read the syllabus and review the semester schedule for topics, assignments, and exams.

- Read the weekly book chapter(s) prior to start of the class.

- Come to class and see me during office hours.

- Participate in classroom discussions.

- Submit homework and project assignments on time.

- Projects – complete them in teams of two.
  - upload solution to Dropbox

- Individual homework assignments.

- Readings in textbook required for class.

- Exams:
  - Midterm exams: Sept. 30th & Oct. 28th given online via quiz
  - Final exam: Monday, Dec. 7th given online via quiz

- Participation in class.

- A software program
  - mediates between hardware resources and user
  - Linux/UNIX Kernel

- A resource allocator.

- A control program.

- Examples: Windows Family, Linux Family, UNIX (Berkely, ATT Bell Labs)

  (On UNIX see
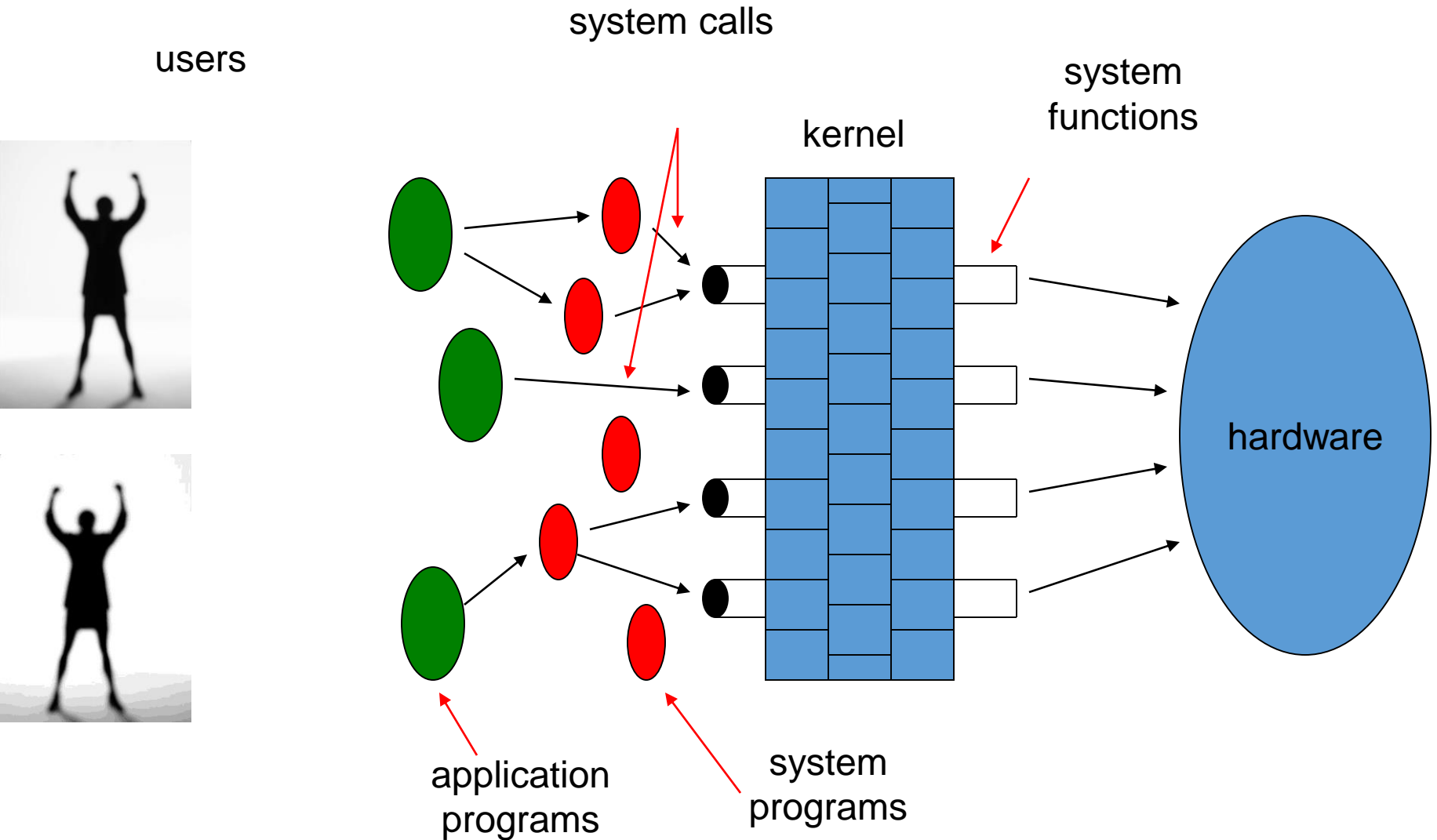   http://www.unix.org/what_is_unix/history_timeline.html)

- Facilitates program execution.

- Supports program development.

- Makes computer system easy to use for users.

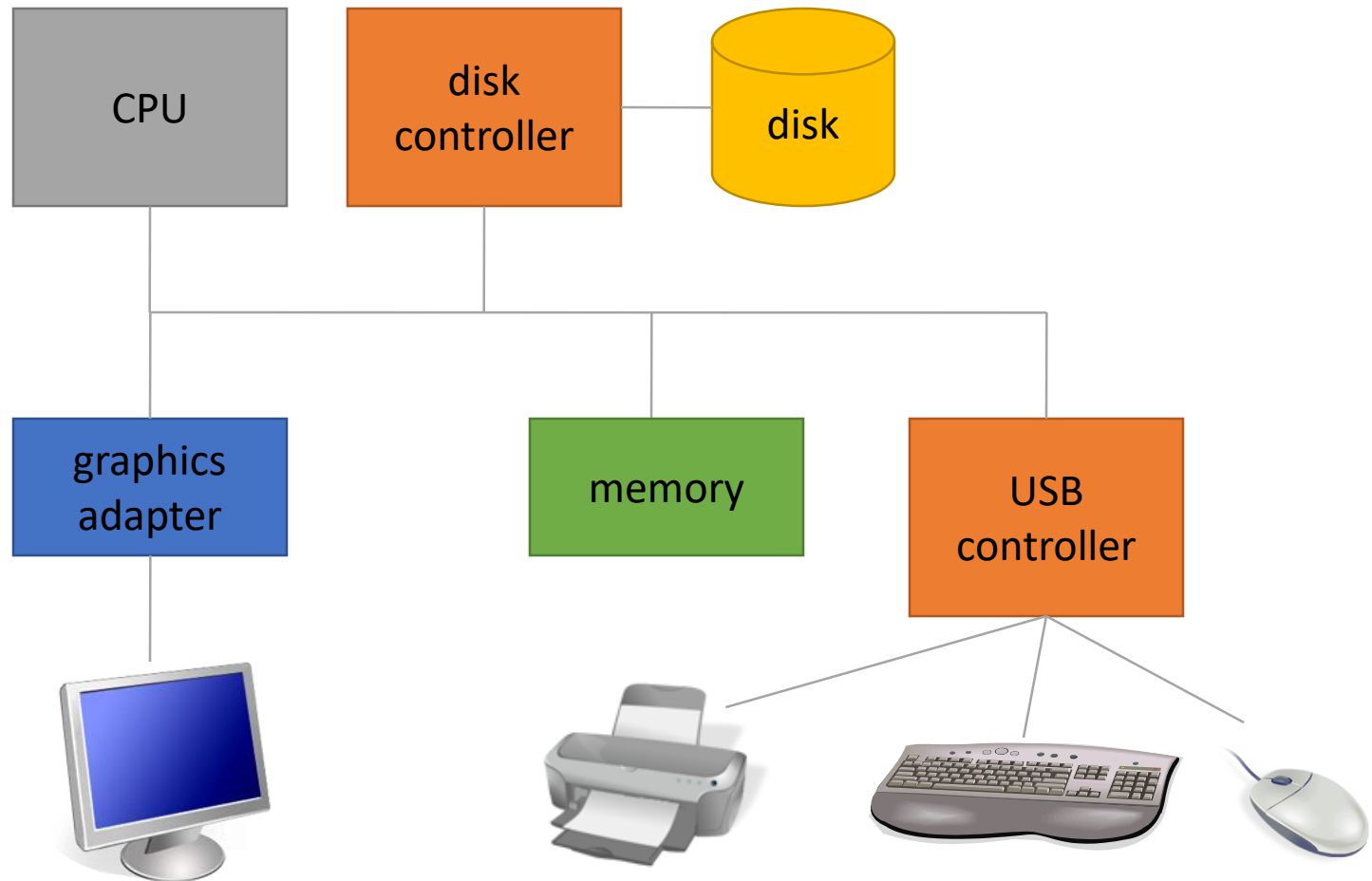- Uses computer resources efficiently.

user

OS

Shell and/or GUI

System Commands

System Calls

Kernel

Hardware

user

user
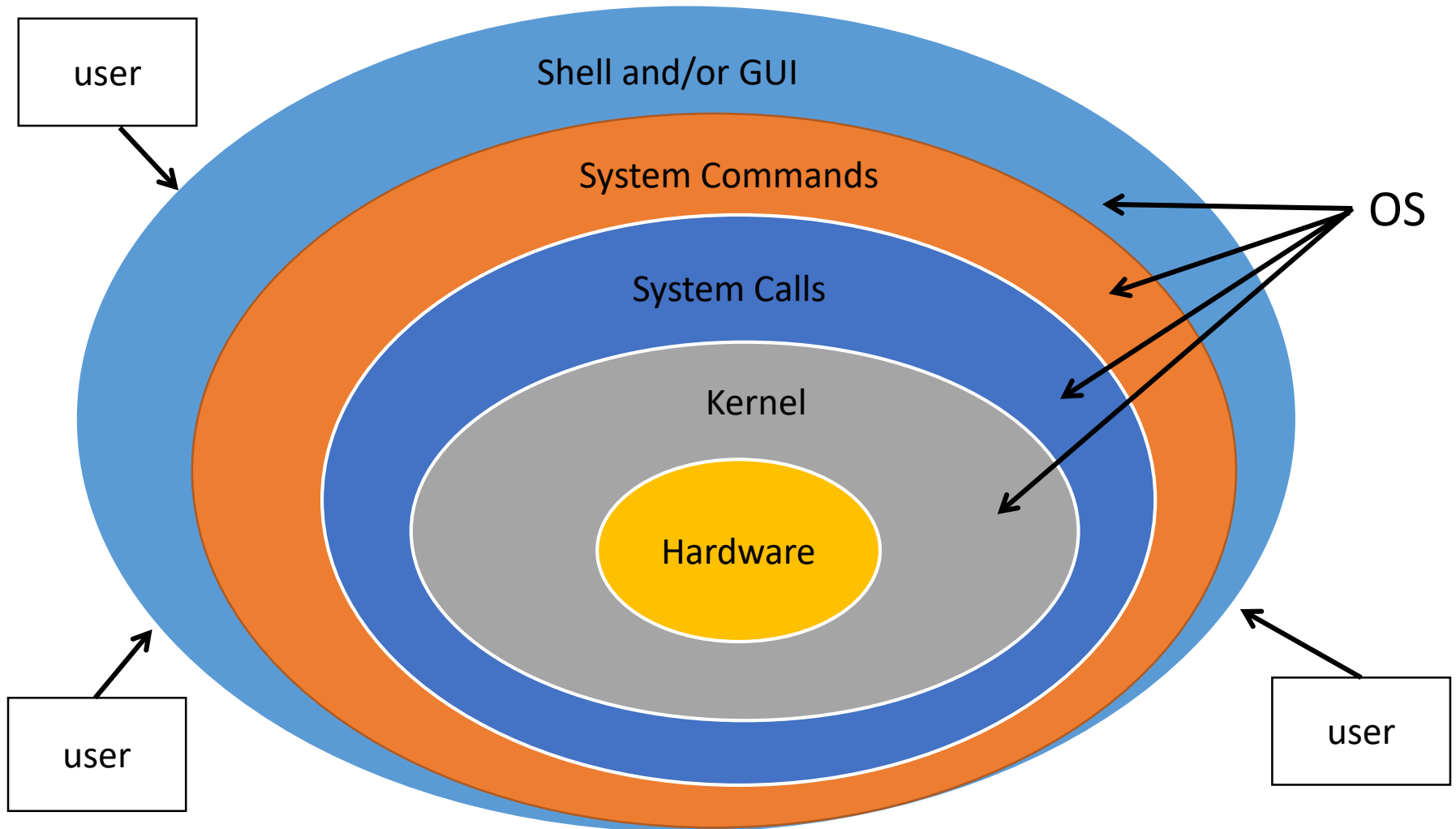
- Hardware:
  - CPU, Memory

- Operating system:
  - device driver
  - interrupt handler
  - scheduler
  - …

- Application programs:
  - use system resources (e.g. libraries)
  - provide interfaces to access system resources
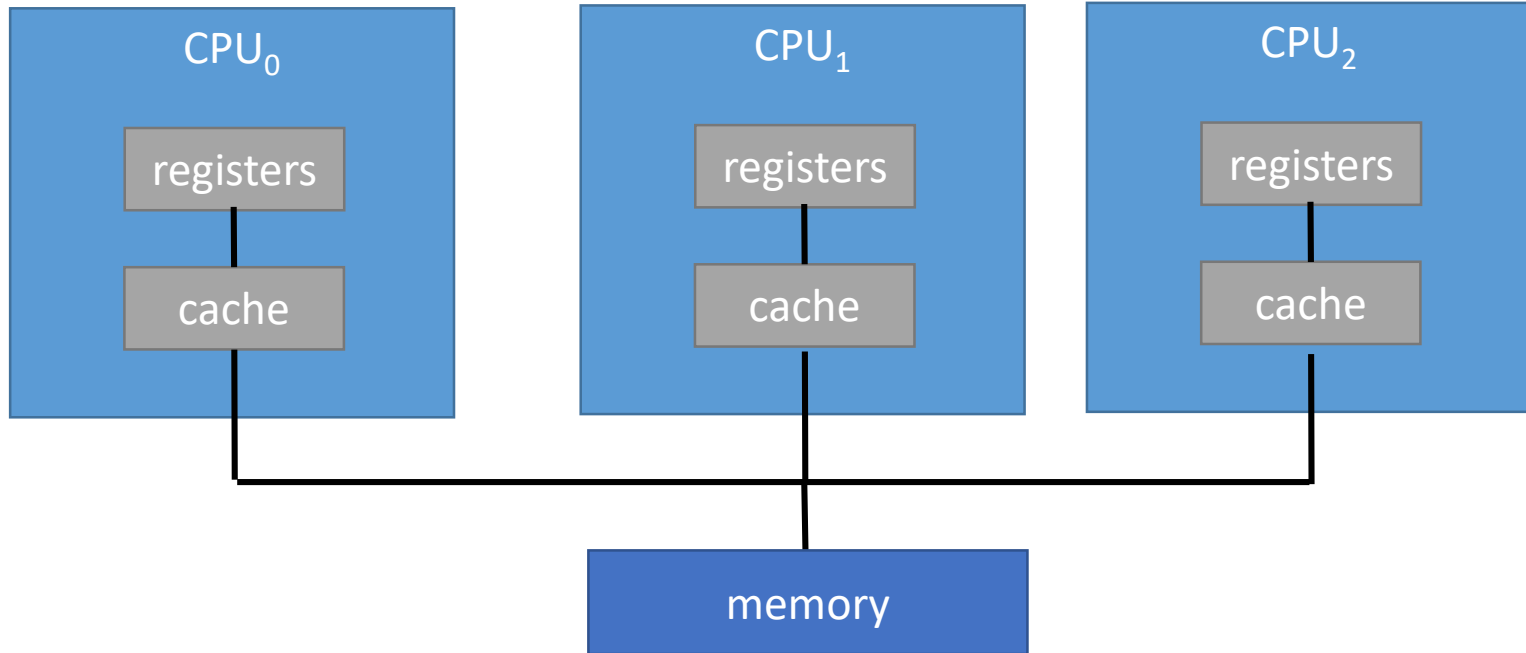
- CPU and I/O devices may operate concurrently.

http://techwiseuk.com/wp-content/uploads/2012/09/CPU.jpg   www.uptechnologysolutions.com

- A device controller operates a device.
  - has local buffer for data exchange
  - uses <u>interrupt</u> to alert CPU
- CPU or DMA perform data exchange.

Multiprocessor System

- Symmetric: all processors execute all functions of the OS
- Asymmetric: master processor controls system, all other processors execute assigned job
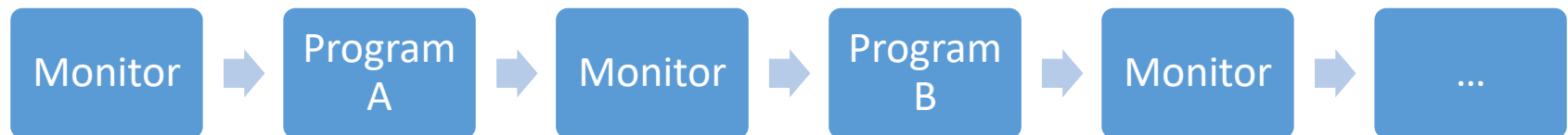
Dual-core Processor

CPU

CPU-core$_0$

registers

cache

CPU-core$_1$

registers
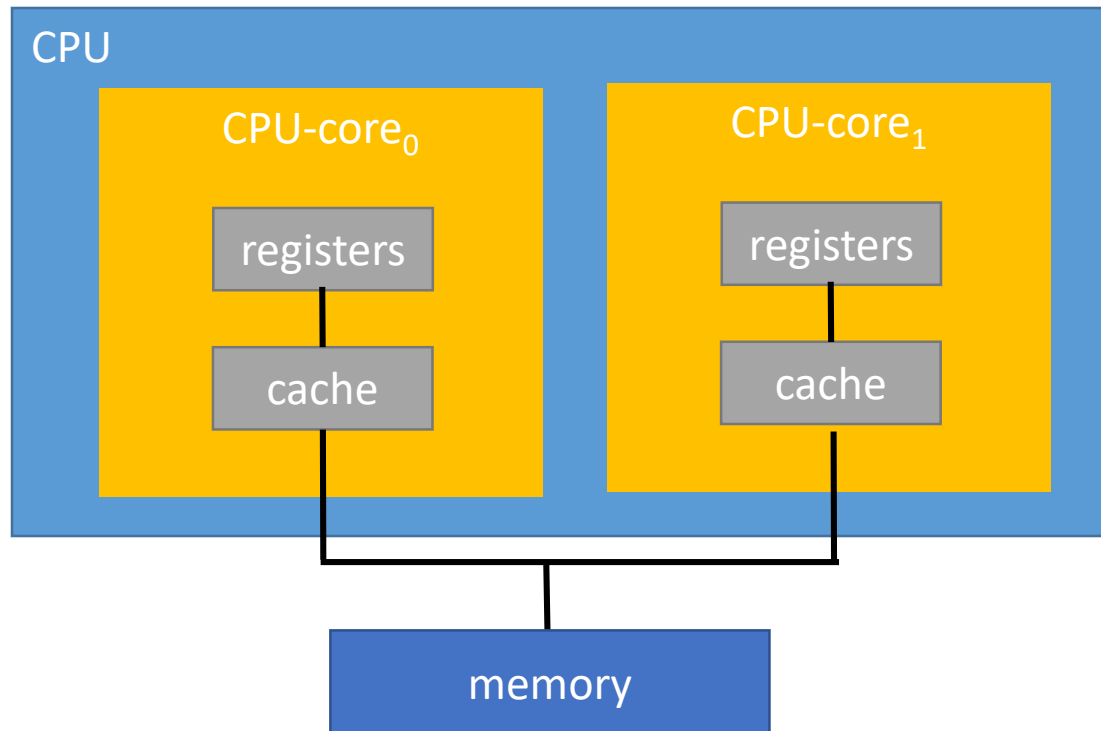
cache

memory

- Cores execute all system functions.
- Cores may share cache.

http://blog.softlayer.com/tag/mainframe/

- Executes a single user program, a.k.a. job, at a given time.

- Perform automatic job sequencing (basic scheduler)

Monitor → Program A → Monitor → Program B → Monitor → …

1. OS loads user program into memory
2. OS starts user program.
3. User program terminates.
4. OS resumes operation.
5. OS loads next user program into memory.
6. …

Operating System
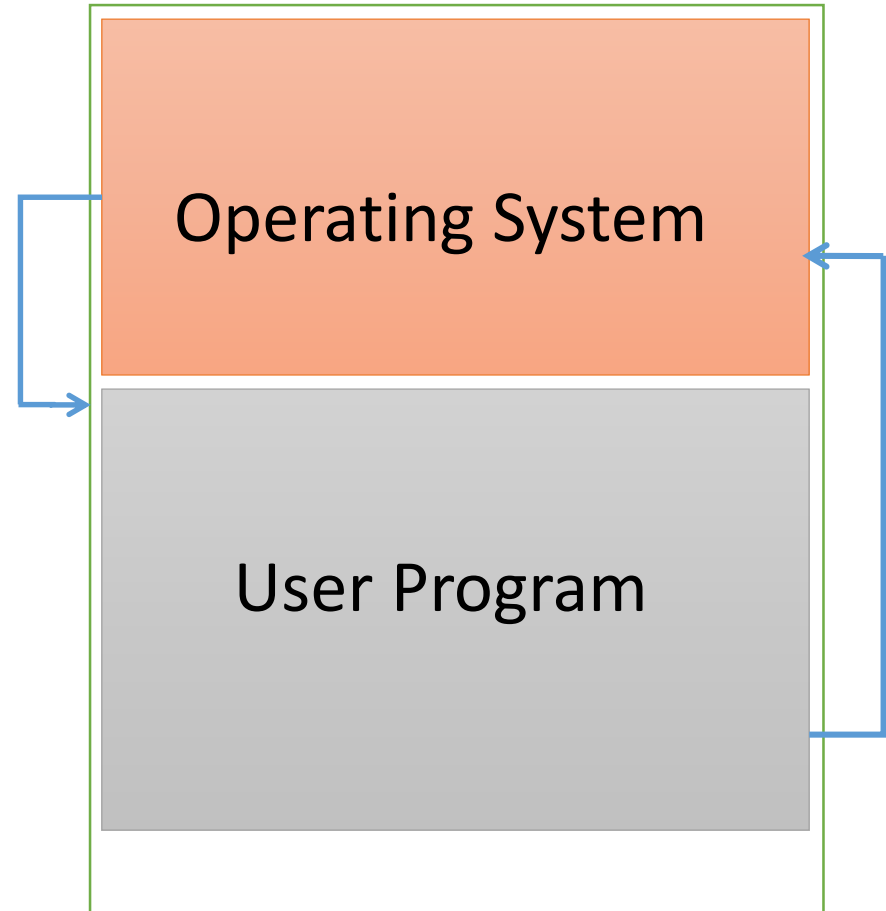
User Program

- Several jobs exist in memory.

- CPU is multiplexed among them.

1. OS selects a job in memory for execution.
2. OS starts user program.
3. User program terminates or performs IO.
4. OS resumes operation.
5. OS selects next job for execution.
6. …

   OS loads new jobs into memory
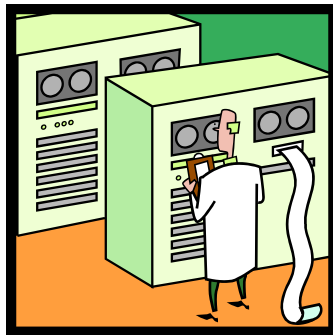
| Operating System |
| Job 1 |
| Job 2 |
| Job 3 |
| Job 4 |
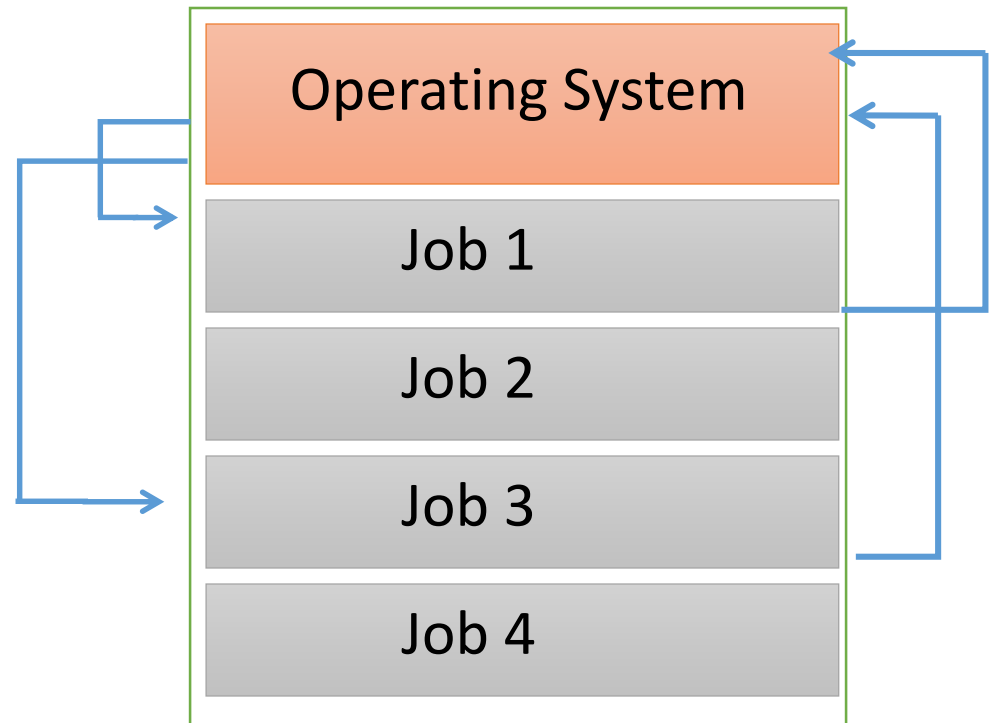
- I/O routine must be provided by OS.

- System must manage memory
  (discussed in details later)

- CPU scheduling
  (discussed in details later)

- OS must manage I/O devices.

- CPU is multiplexed among several jobs.

- Jobs may be swapped in and out of memory.

- Interactive communication between user and system.

one mainframe

many terminals

http://www.youtube.com/watch?v=Anxxe8SdX78
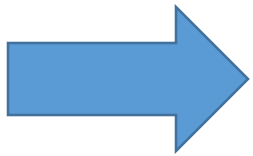
- OS assigns CPU to the next job in memory.
  - program uses IO devices during execution
  - program relinquishes CPU at completion or when a program error occurs

        Loss of Control for OS

- What mechanisms exist for the OS to regain control of the CPU?

  Answer: Software and hardware interrupts.

- # Interrupt transfers control to interrupt service routine
  - interrupt vector vs. interrupt service routines

Hardware interrupts		handler routine 1
	….	
Exceptions	….	handler routine 65
Software interrupts (trap)		
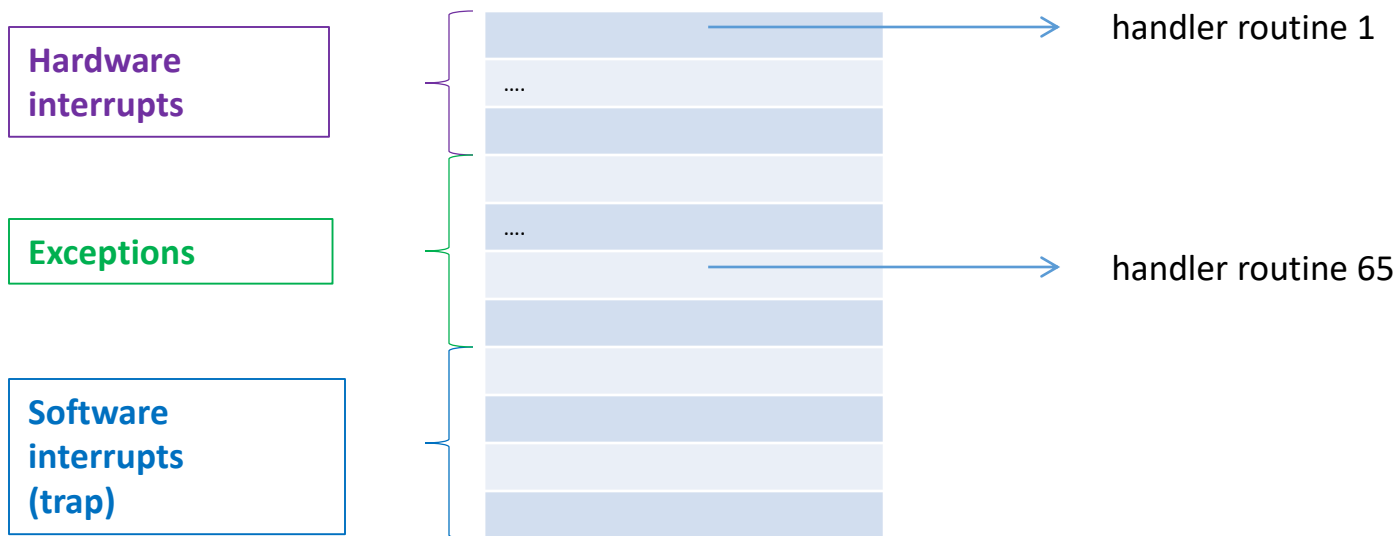
- # When interrupt occurs:
  - save address of instruction being interrupted
  - disallow all or most interrupts

- IO device such as
  - keyboard
  - mouse
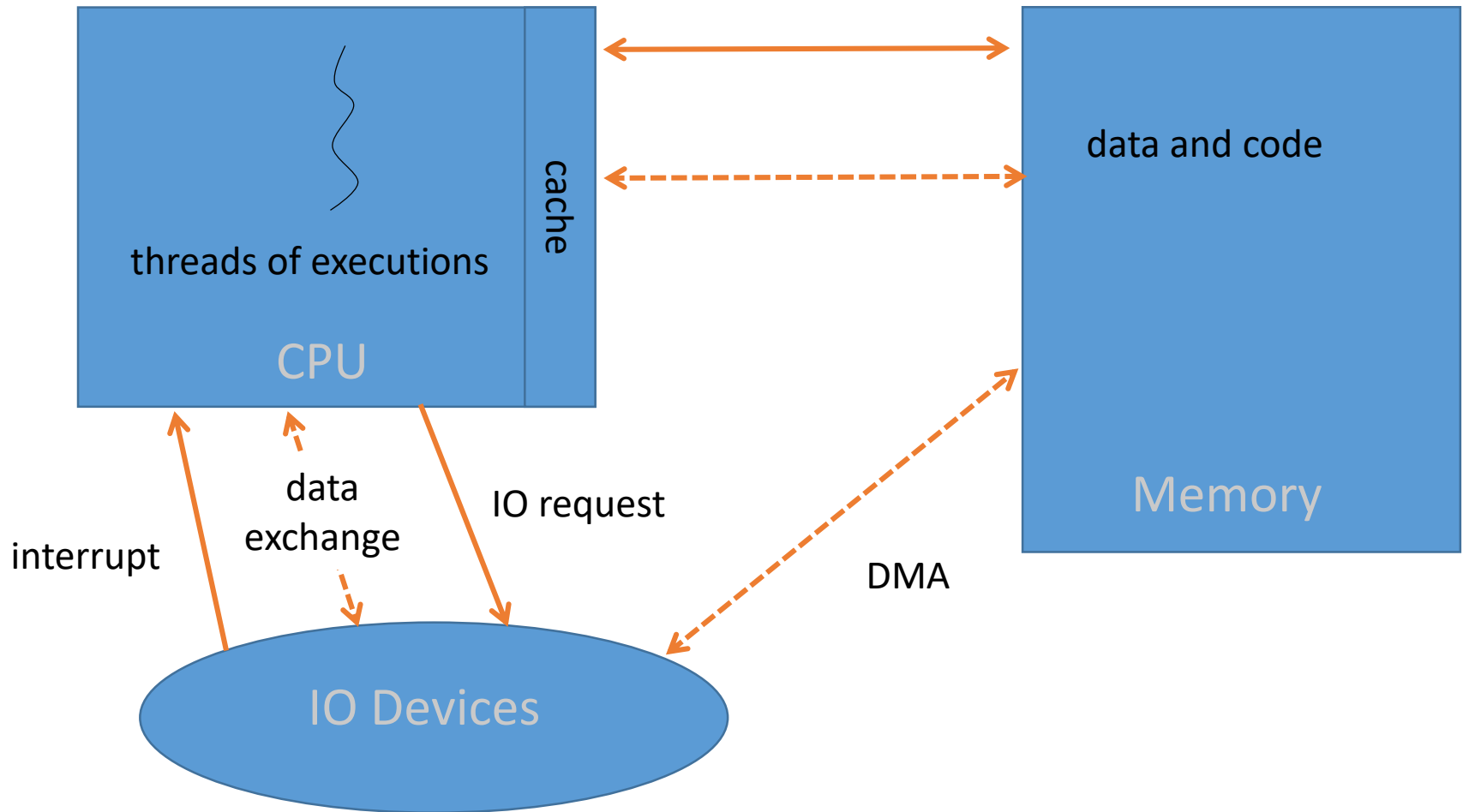 - network card
  - …
- Timer when the time slice has expired to return control back to the OS.
- Programs when
  - a system call is executed
  - a run-time error occurs

L01 Overview

Dual-Mode Protection

- System must ensure that malicious or incorrect code cannot cause other programs to malfunction.

- Hardware provides support for two modes of operation:
  - user mode – execution done on behalf of user
  - kernel mode – execution done on behalf of operating system
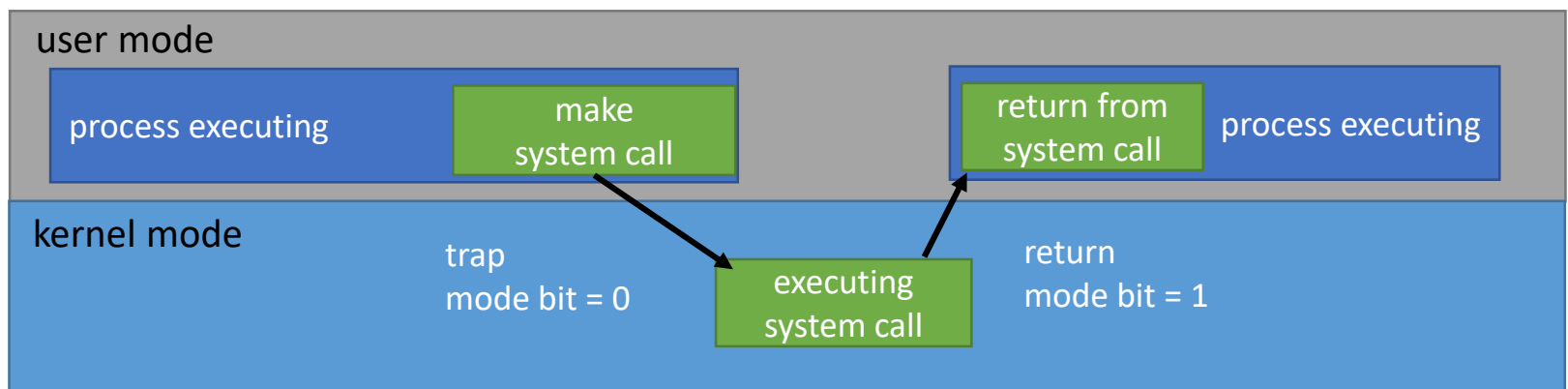
- Mode bit indicates operation mode.
  - 0 for kernel mode
  - 1 for user mode

- An interrupt triggers a change of mode.

- OS sets user mode bit to switch back to user mode.

Privileged instructions only execute in kernel mode.

- All I/O operations are privileged instructions that must be executed in kernel mode.
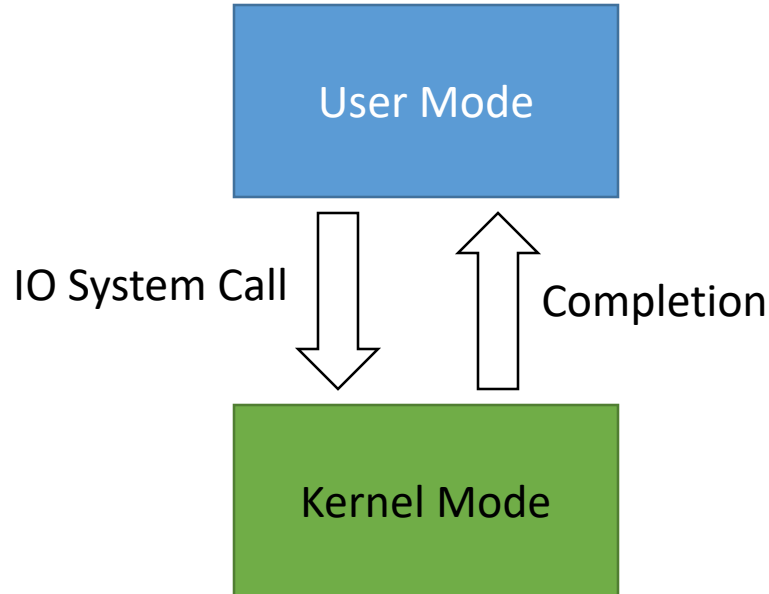
Upon completion, kernel jumps back to user program.

- When more than one program resides in memory, memory protection is needed.

- Two hardware registers specify range of legal address space for program:

MMU

BASE	LIMIT
1280	4096
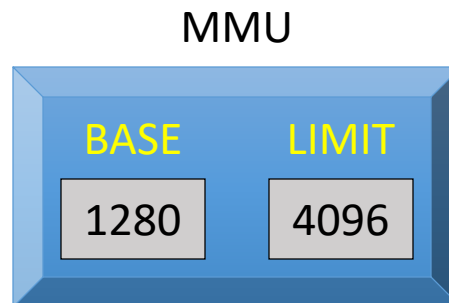
- A program that accesses memory outside the legal address space causes an interrupt.

- CPU must be available to all programs for execution.

- Timer generates interrupt after specified period, to ensure OS regains control.
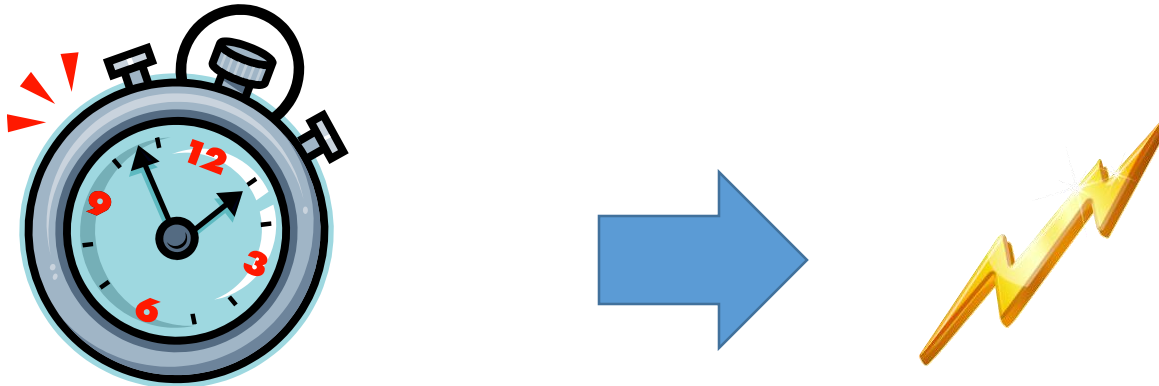
- Timer is used to implement time-sharing systems.

UNIVERSITY *of* WEST FLORIDA
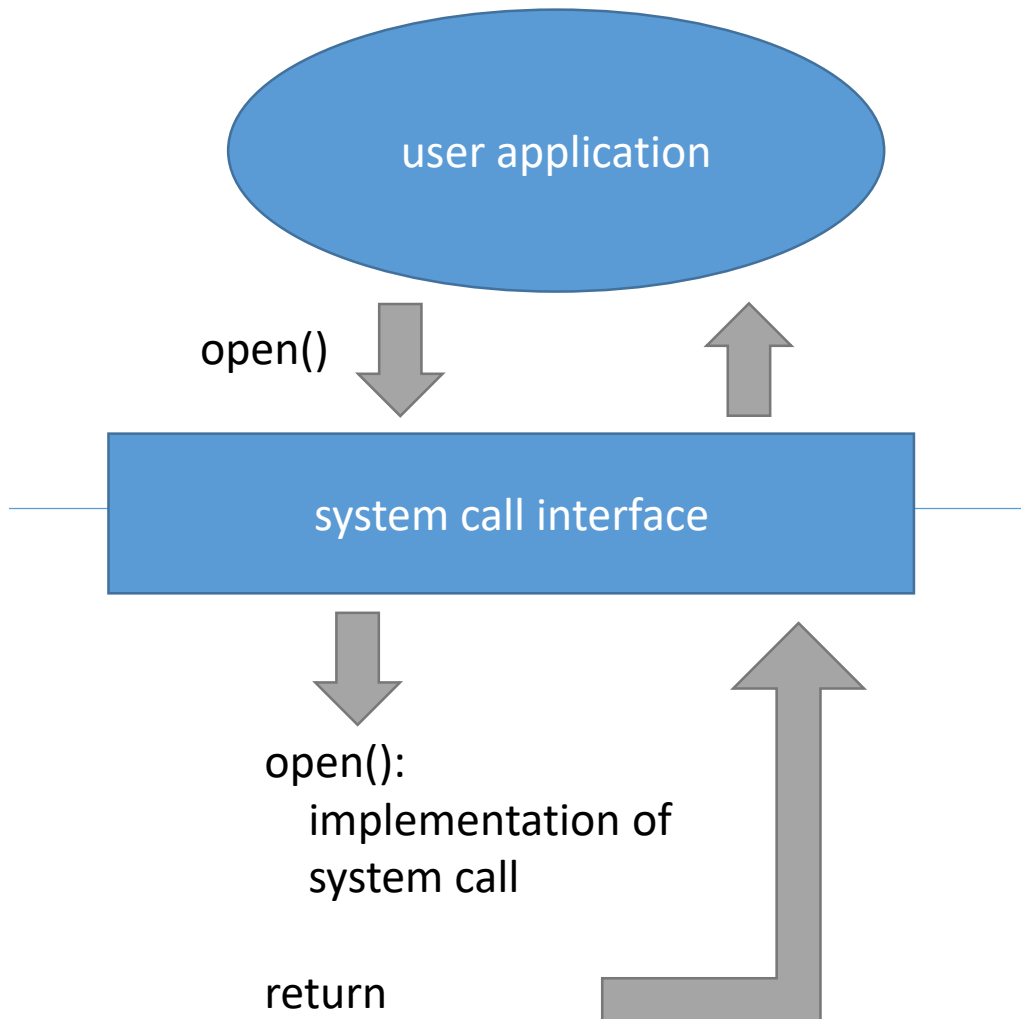
Types of System Call

- process control
- file operations
- device operations
- communication
- …

System Call Examples

- ▸ exit()
- ▸ fork()
- ▸ allocate() / free()
- ▸ open() / close() / write() / read()

user application

open()

system call interface

open():
implementation of
system call

return

Nano	10^{-9}	1/1000000000	Billionth	n
Micro	10^{-6}	1/1000000	Millionth	μ (mu)
Milli	10^{-3}	1/1000	Thousandth	m
Kilo	10^{3}	1000	Thousand	k
Mega	10^{6}	1000000	Million	M
Giga	10^{9}	1000000000	Billion	G
Tera	10^{12}	1000000000000	Trillion	T
Peta	10^{15}	1000000000000000	Quadrillion	P

Computer System (Powers of 2)

Kilobyte	10 bits	2^{10}	1024	KB
Megabyte	20 bits	2^{20}	1048576	MB
Gigabyte	30 bits	2^{30}	1073741824	GB
Terabyte	40 bits	2^{40}	1099511627776	TB
Petabyte	50 bits	2^{50}	1125899906842620	PB

- Website for C/C++ programming

  http://www.cprogramming.com

  http://www.stroustrup.com/C++.html

- C++ Programming in UNIX Systems

  https://www.usna.edu/Users/cs/choi/ic210/lab/l01/lab.html

  https://see.stanford.edu/materials/icsppcs107/08-Unix-Development.pdf

- Internet Forums

  StackOverlow

  CodeGuru

`man`	Display a page of the on-line manual
`ls`	List the contents of a directory
`cd`	Change the current working directory
`pwd`	Display the current working directory
`mkdir`	Make a new directory
`rmdir`	Remove a directory
`rm`	Remove a file or directory
`cp`	Copy a file
`mv`	Move or rename a file or directory

`cat`   Display the contents of a file on the   screen

`more`   Like cat, but a page at a time

`less`   Like more, but more of it

`grep`   Search for a pattern in files

`ps`   List the status of processes in the system

`uname`   Print system information

`gzip`   Compress (decompress) a file

`tar`   Manipulates files in an archive

`lpr`	Print a file
`ispell`	Spell check a text file
`cal`	Display a calendar
`diff`	Compare two files and display the differences
`vi`	One of many text-based editors
`emacs`	One of many GUI-based editors
`gcc`	GNU C compiler
`gdb`	GNU debugger

To run a program w/o options

```
> ls
```

To run a program found in the current directory

```
> ./parse
```

Sending output to a file (redirecting output)

```
> ./parse > parse_out.txt
```

Common argument format (-?)

```
> ./parse -d > parse_debug.txt
```

```
man
man -k search-word
man -s section specific-name
```

Sections

1 User commands

2 System calls

3 Standard library functions

4 Interface libraries

5 Headers

6 Games and demos

7 Devices

8 Maintenance commands and file formats

9 Kernel functions for drivers

n Miscellaneous libraries

```
g++ -g -Wall somefile.cpp -o somefile
```

GNU C++
Compiler

enable all
warning
messages

send the
output to…

produce
debugging
information

source
C++ file

executable
output file

somefile.cpp → g++ → somefile
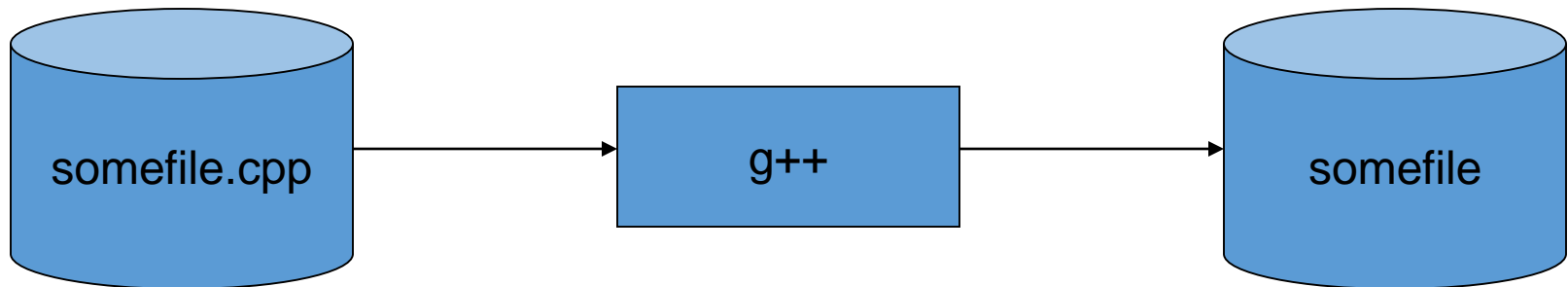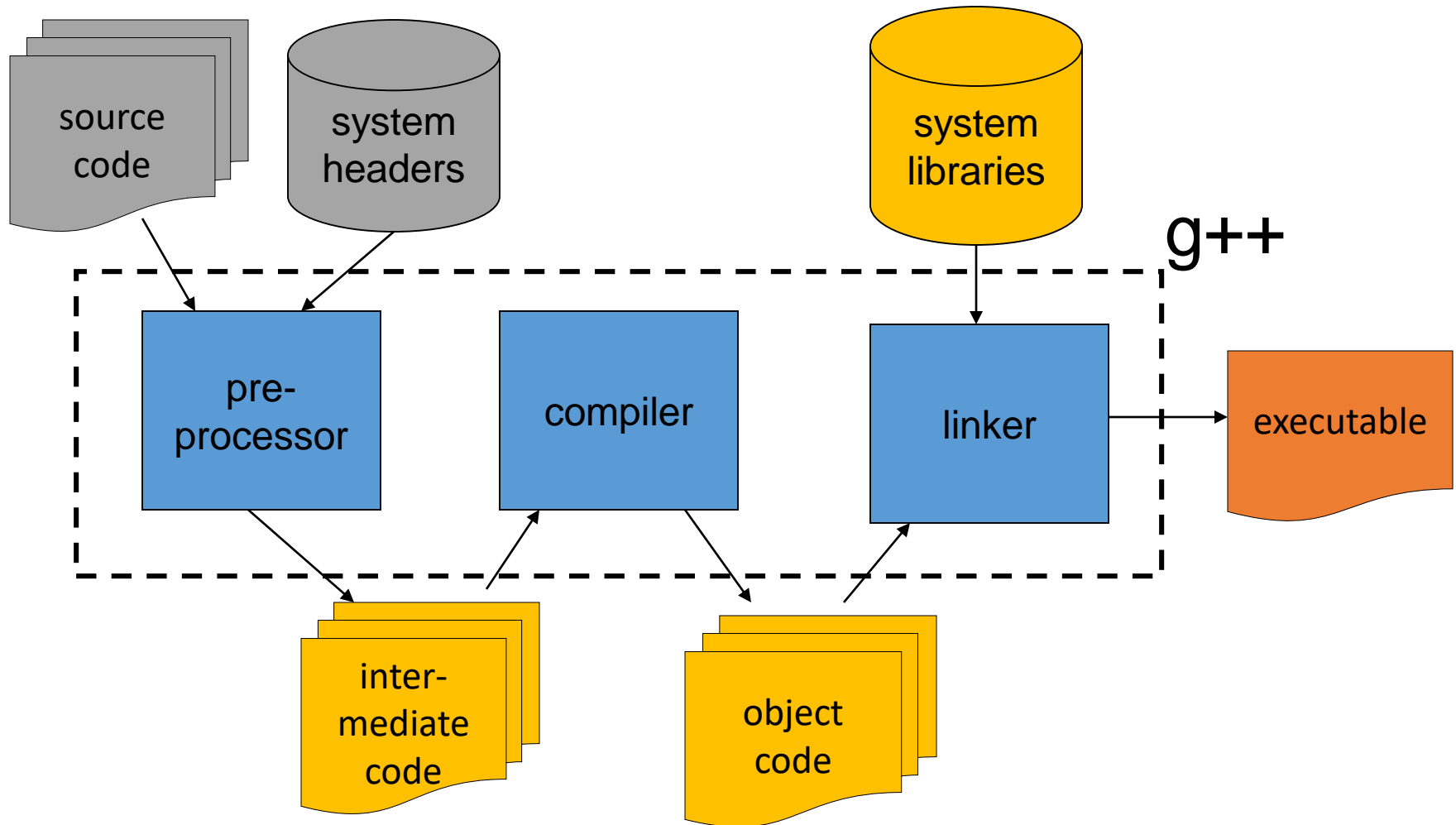
1 Create/edit your program

> `emacs prog1.cpp`

2 Compile your program into an executable

> `g++ -g -Wall prog1.cpp -o prog1`

3 If warnings/errors, go back to step 1

4 Execute program with simple test case

> `./prog1`

5 If errors/output incorrect, go back to step 1

6 Execute program with more complex test case

7 If errors/output incorrect, go to step 1

8 Go to step 6

```
int main( int argc, char **argv )
```

`argc` – number of arguments on the command-line

`argv` – array of arguments (strings) entered on the command-line (arrays coming soon)

```
> ls -l helloMe.cpp
```

argc == 3   argv[1] == "`-l`"

argv[0] == "`ls`"   argv[2] == "`helloMe.cpp`"

`#include` directives

`#define` directives

Namespace directives

Class definitions

`main()` definition

- Automate compile process.
  - why?
  - where?

- System program: *make*
  - a rule interpreter
  - executes instructions to compile and link a program
  - instructions are written in a Makefile

- Rules describe dependencies.
  - helps *make* to decide what must be recompiled
- Format of rules:

```
target : prerequisites
        command
…
```

target: a name of a file or program to be built or an action to be carried out.
command: an instruction to be executed.
prerequisites: a list of files on which the target depends.

1. Compiles a program consisting of a single source code file (*hello.c*) and no header files.

```
hello : hello.cpp
        g++ -o hello hello.cpp
```

2. Compiles a program consisting of two source code files (*hello.c*) and a header file (*sort.h*).

```
test : test.o sort.o
        g++ -o test test.o sort.o

test.o : test.c sort.h
        g++ -c test.c
sort.o : sort.c sort.h
        g++ -c sort.c

clean :
        rm test test.o sort.o
```

3. The previous Makefile with variables to reference all object files.

```
objects = test.o sort.o

test : $(objects)
        g++ -o test $(objects)

test.o : test.cpp sort.h
        g++ -c test.cpp
sort.o : sort.cpp sort.h
        g++ -c sort.cpp

clean :
        rm test $(objects)
```

3. Letting make deduce the commands using an implicit rule for updating .o files.

```
objects = test.o sort.o

test : $(objects)
        g++ -o test $(objects)

test.o : test.cpp sort.h
sort.o : sort.cpp sort.h

.PHONY : clean
clean :
        rm test $(objects)
```

- First-generation operating systems on mainframes consist of a simple monitor program.

- System calls provide access to the kernel's services.

- Hardware and software interrupts transfer control to service routines in the kernel.

- Modern OS offers two mode operation.

- Supported by hardware, the OS provides various protection mechanisms for user programs to run.