Embedded Software

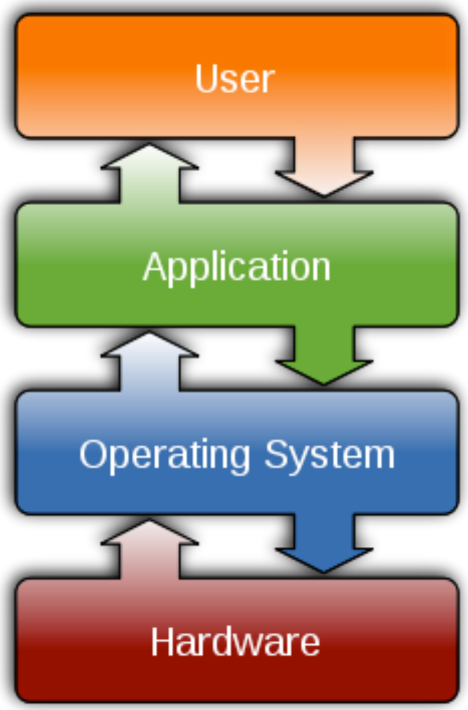
Introduction to operating systems

Agenda

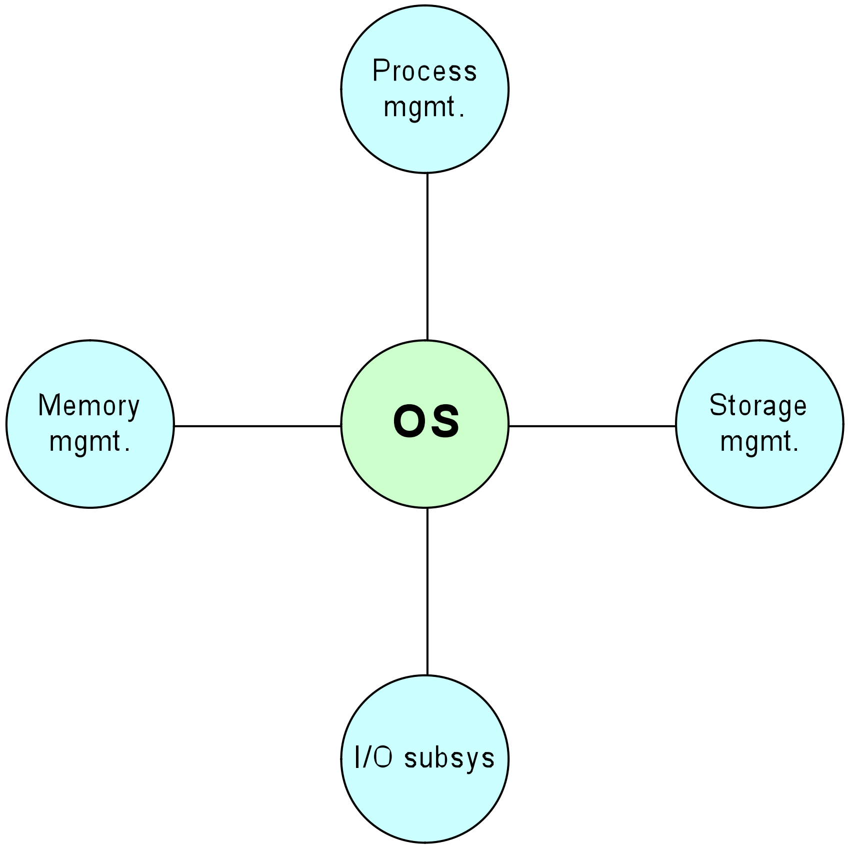
- What is an operating system?
- OS structure
  - Process management
  - Memory and storage mgmt.
  - I/O subsystem
- Operating systems - Real-Time OS's

What is an operating system?

- What is an operating system?
  - Wikipedia: *“An operating system (OS) is software (…) that manages computer hardware resources and provides common services for efficient execution of various application software.”*

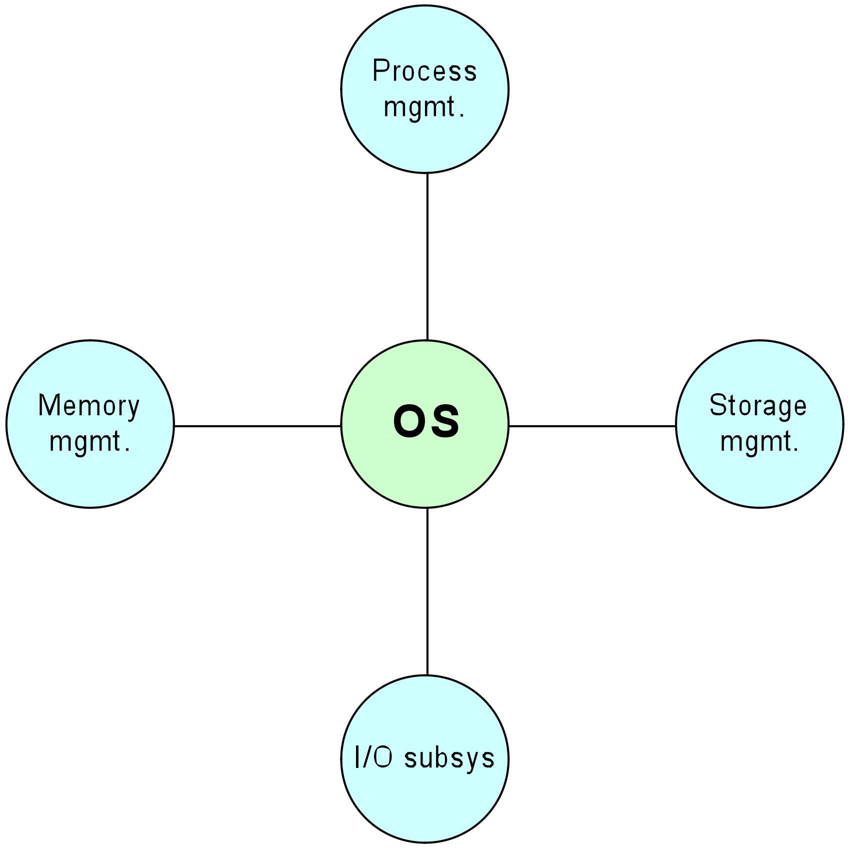


OS structure



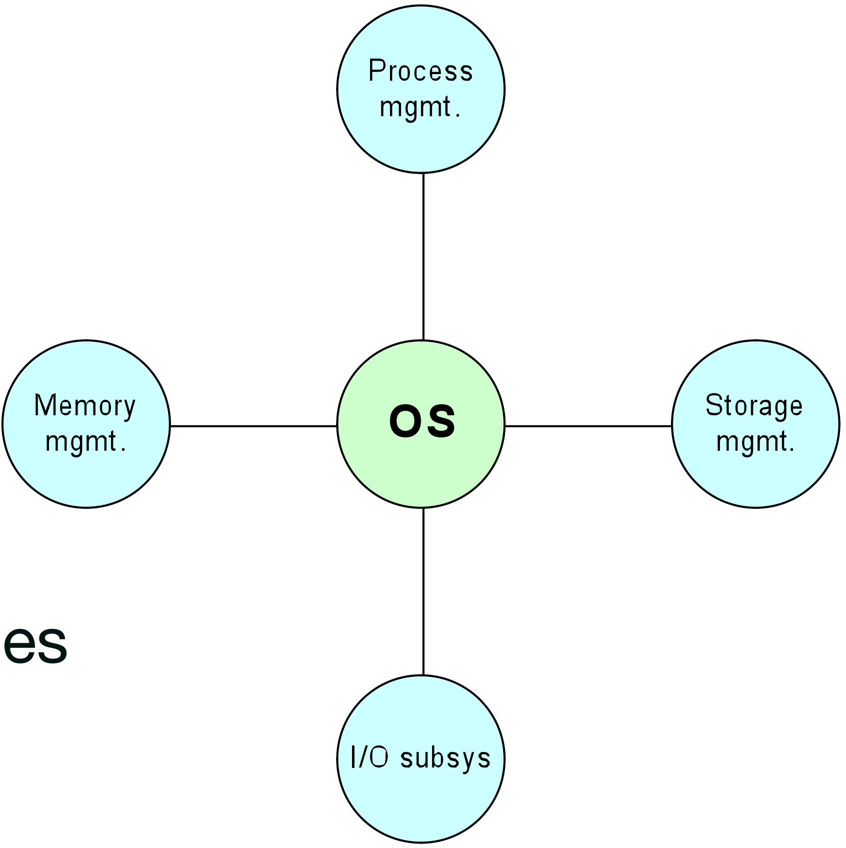
OS structure

- Many computer types – many OS designs
  - Mainframe OSs are optimized for HW utilization
  - Desktop OSs are optimized for generality
  - Embedded OSs are optimized for efficiency, size, safety, speed, low power
  - …



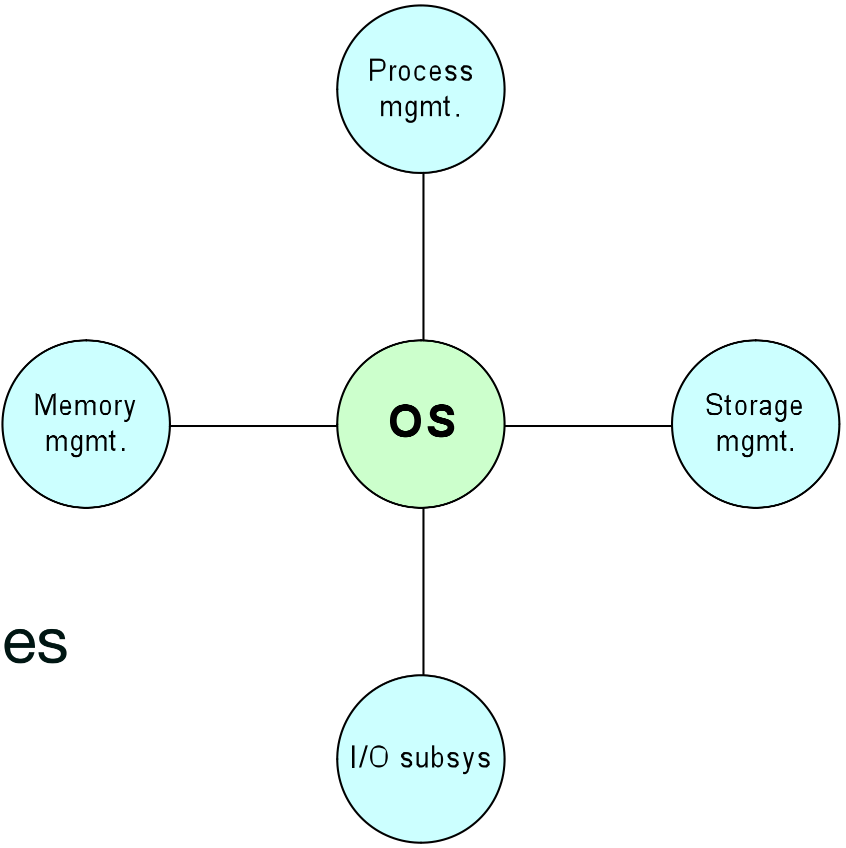
OS structure

- Many computer types – many OS designs
  - Mainframe OSs are optimized for HW utilization
  - Desktop OSs are optimized for generality
  - Embedded OSs are optimized for efficiency, size, safety, speed, low power
  - …
- Some commonalities, though:
  - Process management – handles multiprogramming and keep the CPU busy
  - Memory management – (de)allocation and process swapping
  - Storage management – persistent storage and cache
  - I/O subsystem management – manage I/O devices

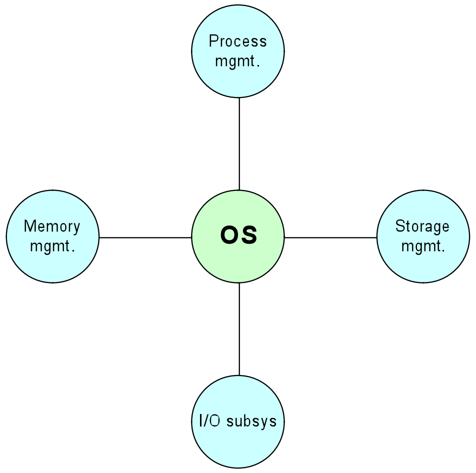


OS structure

- Many computer types – many OS designs
  - Mainframe OSs are optimized for HW utilization
  - Desktop OSs are optimized for generality
  - Embedded OSs are optimized for efficiency, size, safety, speed, low power
  - …
- Some commonalities, though:
  - Process management – handles multiprogramming and keep the CPU busy
  - Memory management – (de)allocation and process swapping
  - Storage management – persistent storage and cache
  - I/O subsystem management – manage I/O devices
- Let’s take a look at process management

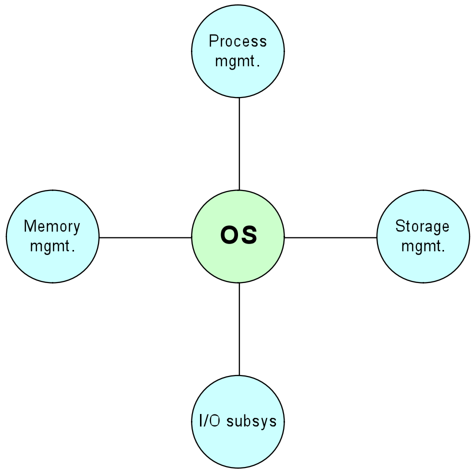


OS structure - Process management



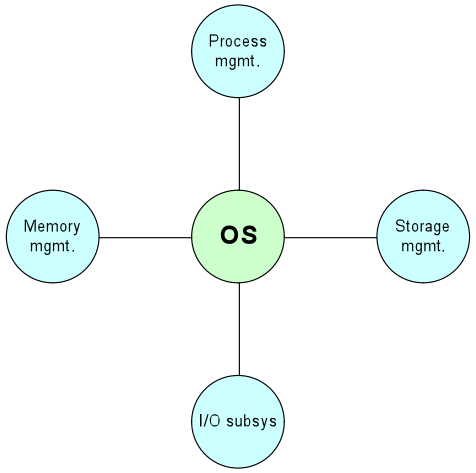
OS structure - Process management

- ***What is a process? Is it a program?***
  - No - a process is a program in execution



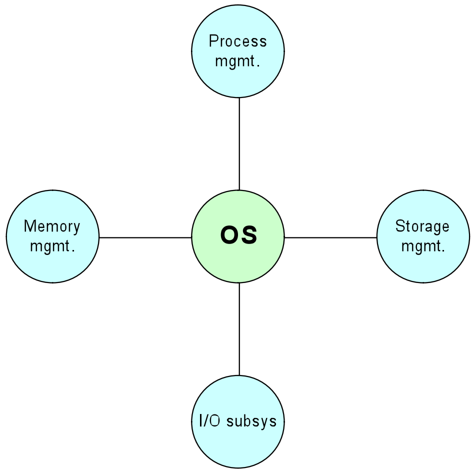
OS structure - Process management

- ***What is a process? Is it a program?***
  - No - a process is a program in execution



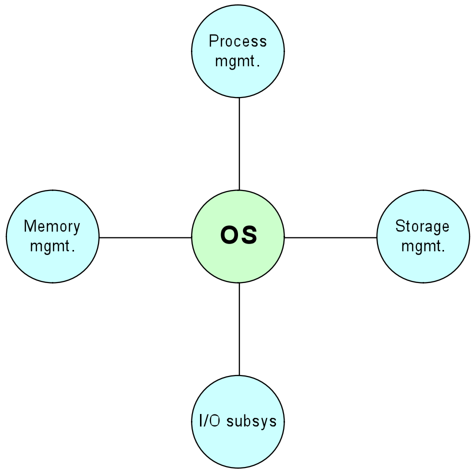
OS structure - Process management

- ***What is a process? Is it a program?***
  - No - a process is a program in execution



- ***How many processes can run at a time?***
  - There can be many processes that want to run, but only one per CPU that actually runs

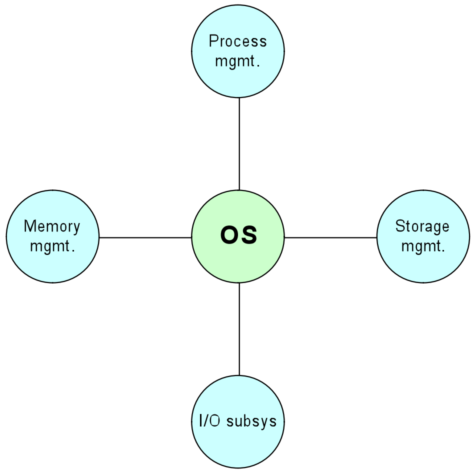
OS structure - Process management



- ***What is a process? Is it a program?***
  - No - a process is a program in execution

- ***How many processes can run at a time?***
  - There can be many processes that want to run, but only one per CPU that actually runs

OS structure - Process management



- ***What is a process? Is it a program?***
  - No - a process is a program in execution

- ***How many processes can run at a time?***
  - There can be many processes that want to run, but only one per CPU that actually runs

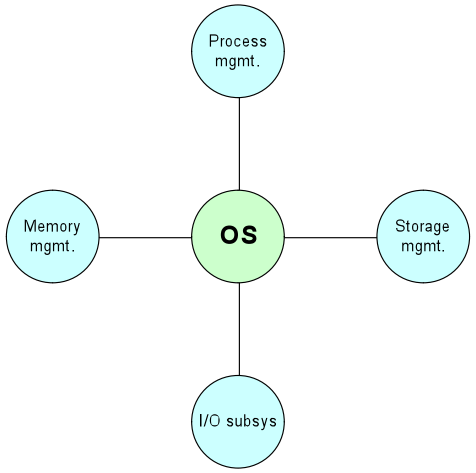
- The OS manages processes
  - Creates, deletes, and allocates resources for them
  - Swaps them in and out of memory
  - Suspends and resumes them
  - Provides mechanisms for synchronization and communication between processes

Process management – why?

- Processes either compute or perform device I/O

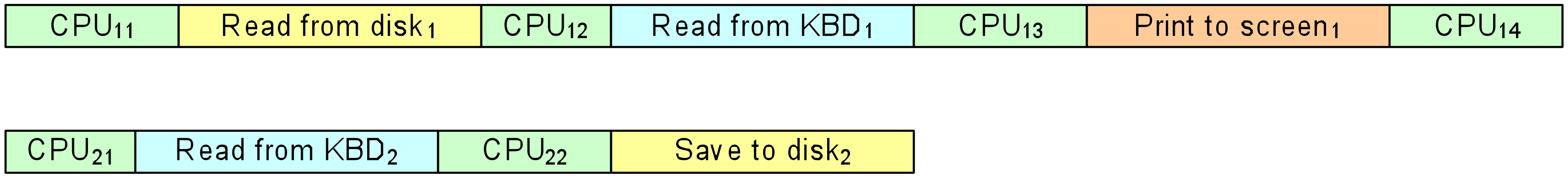
- ***What does a process do while it performs I/O?***
  - It must wait for I/O to complete before it can resume

- ***What should the system do meanwhile?***
  - Without process management: CPU idles
  - With process management: Switch to another ready process



Process management - example

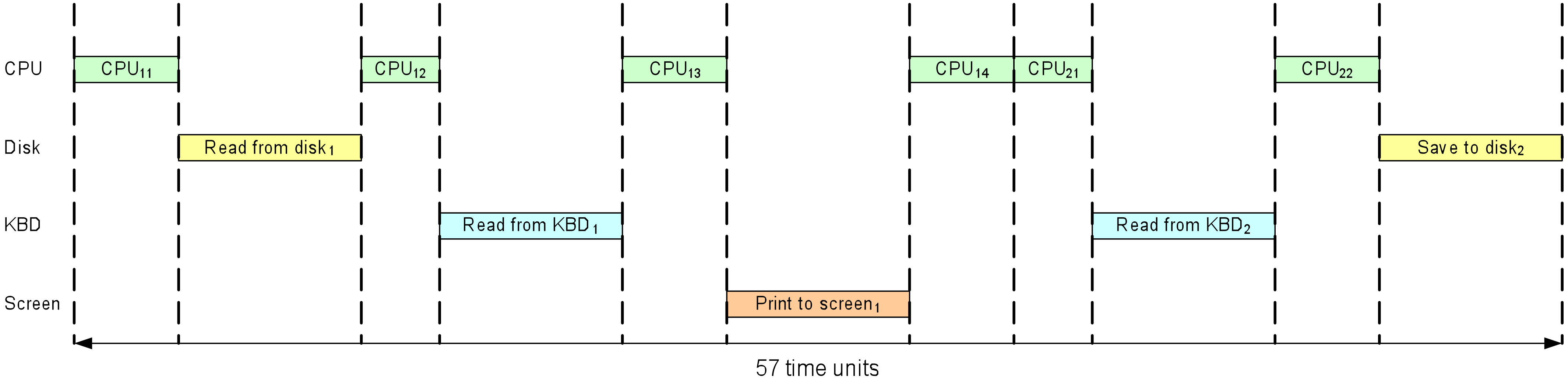
- Consider two tasks:



CPU$_{11}$ | Read from disk$_1$ | CPU$_{12}$ | Read from KBD$_1$ | CPU$_{13}$ | Print to screen$_1$ | CPU$_{14}$

CPU$_{21}$ | Read from KBD$_2$ | CPU$_{22}$ | Save to disk$_2$

- ***Scheduling without resource management (batch processing)?***



CPU: CPU$_{11}$, CPU$_{12}$, CPU$_{13}$, CPU$_{14}$, CPU$_{21}$, CPU$_{22}$

Disk: Read from disk$_1$, Save to disk$_2$

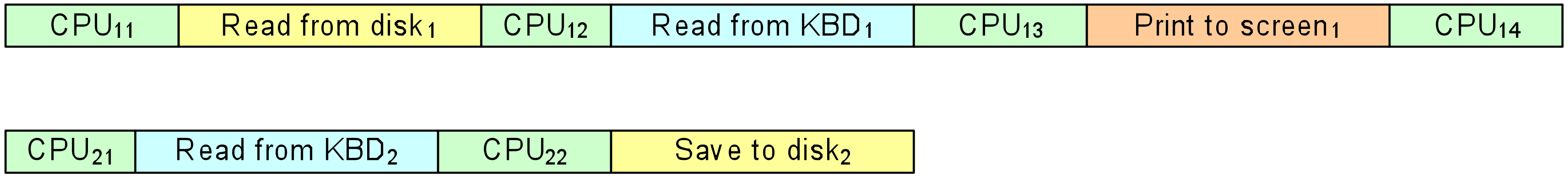
KBD: Read from KBD$_1$, Read from KBD$_2$

Screen: Print to screen$_1$

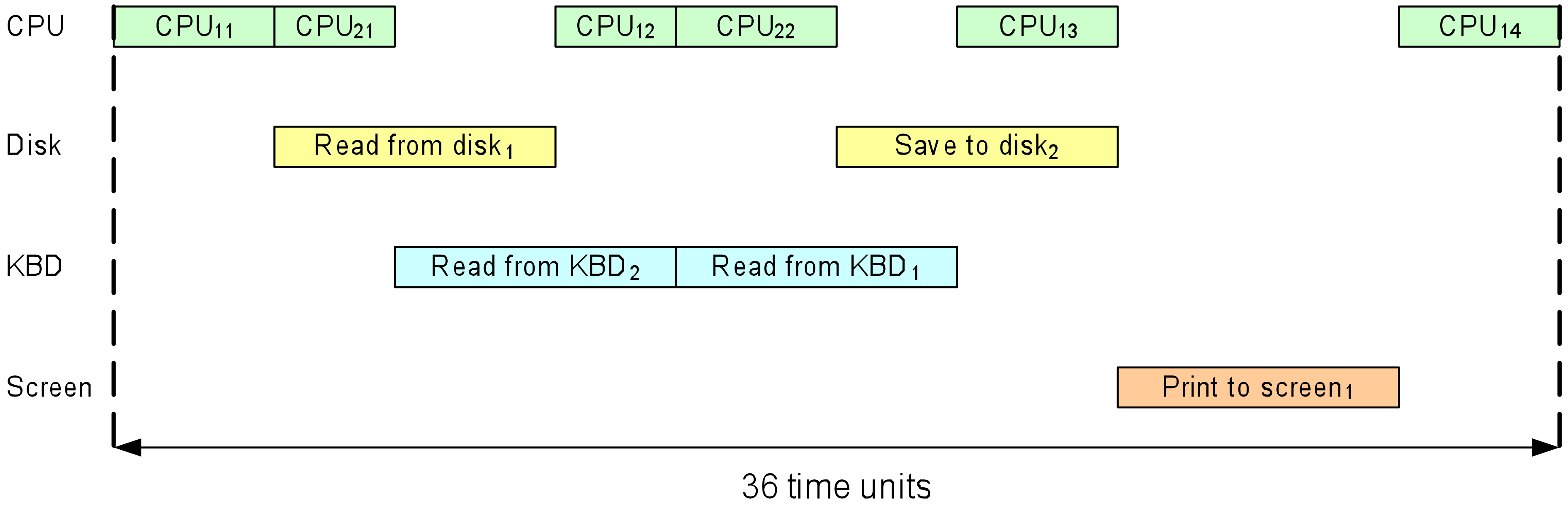
57 time units

Process management - example

- Consider two tasks:



- ***Scheduling <u>with</u> resource management?***



Process management - protection

Process management - protection

- Consider an “evil” process – what damage could it do?
  - Destroy, eavesdrop on, change other processes
  - Destroy OS
  - Destroy files and HW

Process management - protection

- Consider an “evil” process – what damage could it do?
  - Destroy, eavesdrop on, change other processes
  - Destroy OS
  - Destroy files and HW

Process management - protection

- Consider an “evil” process – what damage could it do?
  - Destroy, eavesdrop on, change other processes
  - Destroy OS
  - Destroy files and HW

- The OS guards against this using dual-mode operation (MODE bit in CPU)
  - Applications run in user mode (AKA restricted mode)
  - The OS kernel runs in kernel mode (AKA protected, privileged, supervisor mode)

Process management - protection

- Consider an “evil” process – what damage could it do?
  - Destroy, eavesdrop on, change other processes
  - Destroy OS
  - Destroy files and HW

- The OS guards against this using dual-mode operation (MODE bit in CPU)
  - Applications run in user mode (AKA restricted mode)
  - The OS kernel runs in kernel mode (AKA protected, privileged, supervisor mode)

Process management - protection

- Consider an “evil” process – what damage could it do?
  - Destroy, eavesdrop on, change other processes
  - Destroy OS
  - Destroy files and HW

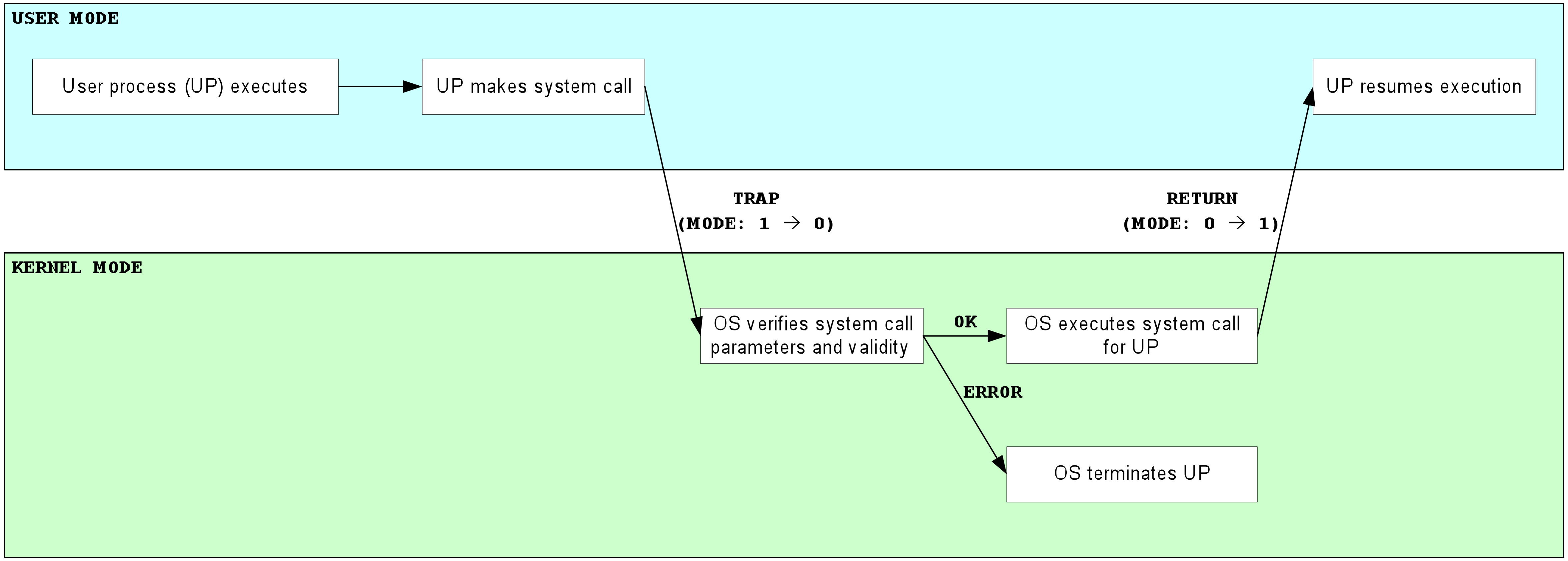
- The OS guards against this using dual-mode operation (MODE bit in CPU)
  - Applications run in user mode (AKA restricted mode)
  - The OS kernel runs in kernel mode (AKA protected, privileged, supervisor mode)

- Potentially dangerous operations (I/O, IPC, …) can only be done via privileged instructions
  - Restricted instructions – user and kernel mode
  - Privileged instructions – kernel mode only

Dual-mode operation – system calls

- When processes need to perform I/O, it does so via the OS via well-defined ***system calls*** (version 2.6.35: 337 different syscalls)

- The OS (which is in kernel mode) verifies the system call and its parameters



Dual-mode operation – system calls

- How often are system calls made?

Dual-mode operation – system calls

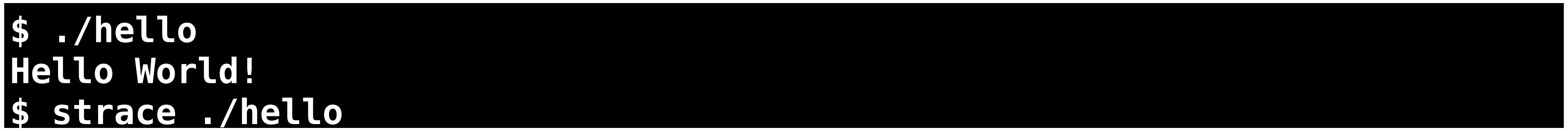
- How often are system calls made?

```
$ ./hello
Hello World!
```

```
$
```

Dual-mode operation – system calls

- How often are system calls made?

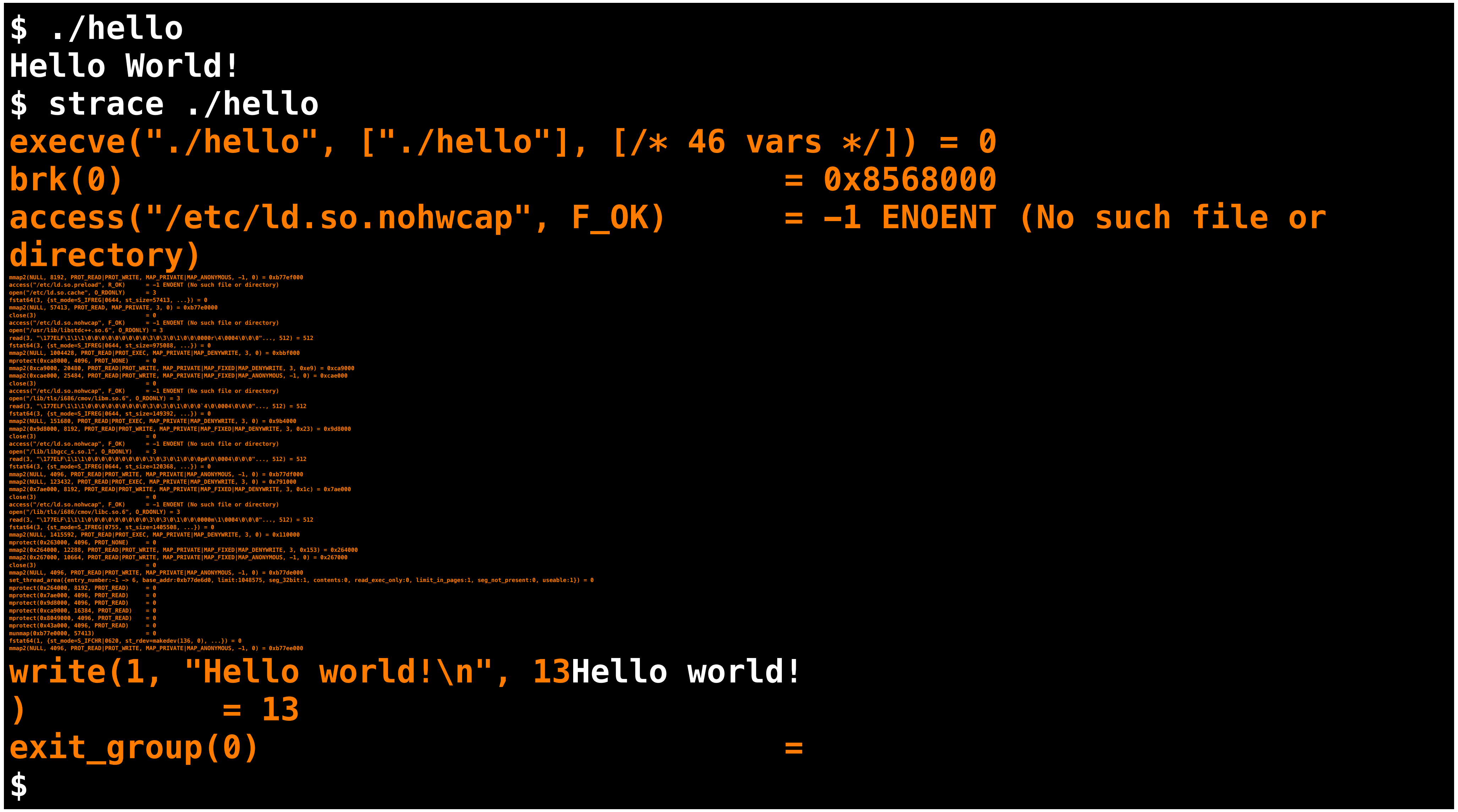


```
$ ./hello
Hello World!
$ strace ./hello
```

```
$
```

Dual-mode operation – system calls

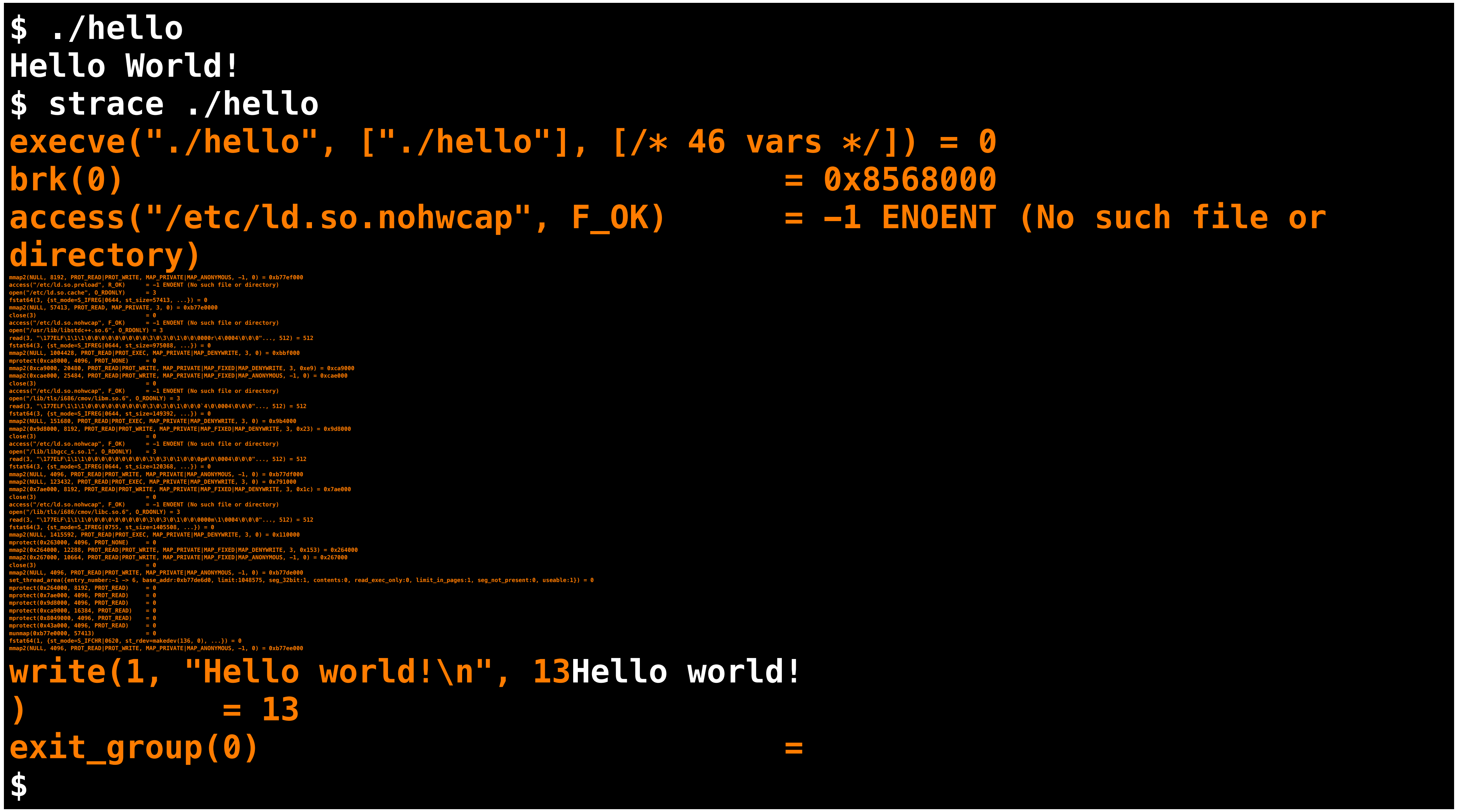
- How often are system calls made?



```
$ ./hello
Hello World!
$ strace ./hello
execve("./hello", ["./hello"], [/* 46 vars */]) = 0
brk(0)                                  = 0x8568000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or
directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb77ef000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=57413, ...}) = 0
mmap2(NULL, 57413, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb77e0000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/usr/lib/libstdc++.so.6", O_RDONLY) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0000r\4\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0644, st_size=975088, ...}) = 0
mmap2(NULL, 1004428, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xbbf000
mprotect(0xca8000, 4096, PROT_NONE)     = 0
mmap2(0xca9000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe9) = 0xca9000
mmap2(0xcae000, 25484, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xcae000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/tls/i686/cmov/libm.so.6", O_RDONLY) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0`4\0\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0644, st_size=149392, ...}) = 0
mmap2(NULL, 151680, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x9b4000
mmap2(0x9d8000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x23) = 0x9d8000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libgcc_s.so.1", O_RDONLY)    = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0p#\0\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0644, st_size=120368, ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb77df000
mmap2(NULL, 123432, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x791000
mmap2(0x7ae000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c) = 0x7ae000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/tls/i686/cmov/libc.so.6", O_RDONLY) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0000m\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1405508, ...}) = 0
mmap2(NULL, 1415592, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x110000
mprotect(0x263000, 4096, PROT_NONE)     = 0
mmap2(0x264000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x153) = 0x264000
mmap2(0x267000, 10664, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x267000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb77de000
set_thread_area({entry_number:-1 -> 6, base_addr:0xb77de6d0, limit:1048575, seg_32bit:1, contents:0, read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
mprotect(0x264000, 8192, PROT_READ)     = 0
mprotect(0x7ae000, 4096, PROT_READ)     = 0
mprotect(0x9d8000, 4096, PROT_READ)     = 0
mprotect(0xca9000, 16384, PROT_READ)    = 0
mprotect(0x8049000, 4096, PROT_READ)    = 0
mprotect(0x43a000, 4096, PROT_READ)     = 0
munmap(0xb77e0000, 57413)               = 0
fstat64(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb77ee000
write(1, "Hello world!\n", 13Hello world!
)          = 13
exit_group(0)                           =
$
```

Dual-mode operation – system calls

- How often are system calls made?

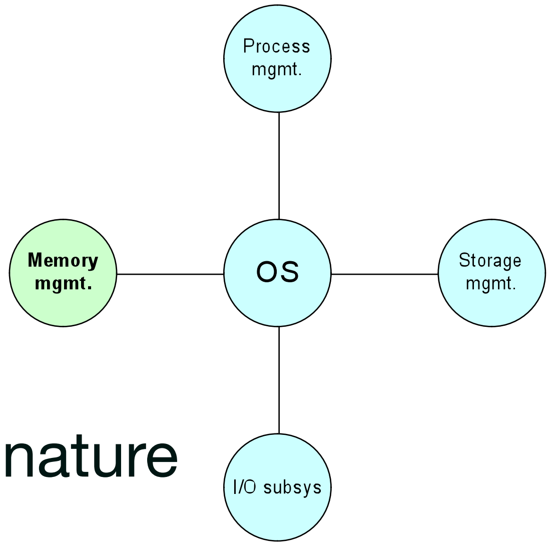


```
$ ./hello
Hello World!
$ strace ./hello
execve("./hello", ["./hello"], [/* 46 vars */]) = 0
brk(0)                                   = 0x8568000
access("/etc/ld.so.nohwcap", F_OK)       = -1 ENOENT (No such file or
directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb77ef000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=57413, ...}) = 0
mmap2(NULL, 57413, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb77e0000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/usr/lib/libstdc++.so.6", O_RDONLY) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0000r\4\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0644, st_size=975088, ...}) = 0
mmap2(NULL, 1004428, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xbbf000
mprotect(0xca8000, 4096, PROT_NONE)     = 0
mmap2(0xca9000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe9) = 0xca9000
mmap2(0xcae000, 25484, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xcae000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/tls/i686/cmov/libm.so.6", O_RDONLY) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0`4\0\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0644, st_size=149392, ...}) = 0
mmap2(NULL, 151680, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x9b4000
mmap2(0x9d8000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x23) = 0x9d8000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libgcc_s.so.1", O_RDONLY)    = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0p#\0\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0644, st_size=120368, ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb77df000
mmap2(NULL, 123432, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x791000
mmap2(0x7ae000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c) = 0x7ae000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/tls/i686/cmov/libc.so.6", O_RDONLY) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0000m\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1405508, ...}) = 0
mmap2(NULL, 1415592, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x110000
mprotect(0x263000, 4096, PROT_NONE)     = 0
mmap2(0x264000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x153) = 0x264000
mmap2(0x267000, 10664, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x267000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb77de000
set_thread_area({entry_number:-1 -> 6, base_addr:0xb77de6d0, limit:1048575, seg_32bit:1, contents:0, read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
mprotect(0x264000, 8192, PROT_READ)     = 0
mprotect(0x7ae000, 4096, PROT_READ)     = 0
mprotect(0x9d8000, 4096, PROT_READ)     = 0
mprotect(0xca9000, 16384, PROT_READ)    = 0
mprotect(0x8049000, 4096, PROT_READ)    = 0
mprotect(0x43a000, 4096, PROT_READ)     = 0
munmap(0xb77e0000, 57413)               = 0
fstat64(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb77ee000
write(1, "Hello world!\n", 13Hello world!
)          = 13
exit_group(0)                            =
$
```

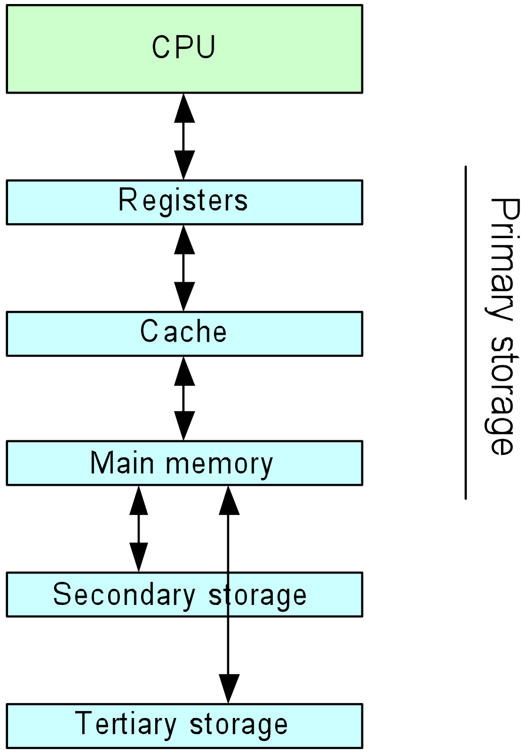
56 system calls

OS Structure - Memory and storage mgmt.

- Memory management
  - Keep track of several processes in memory at a time
  - Decide which (parts of) processes to move in and out of memory
    - Many different algorithms depending on hardware and the OS' nature
  - Allocate and deallocate memory as necessary



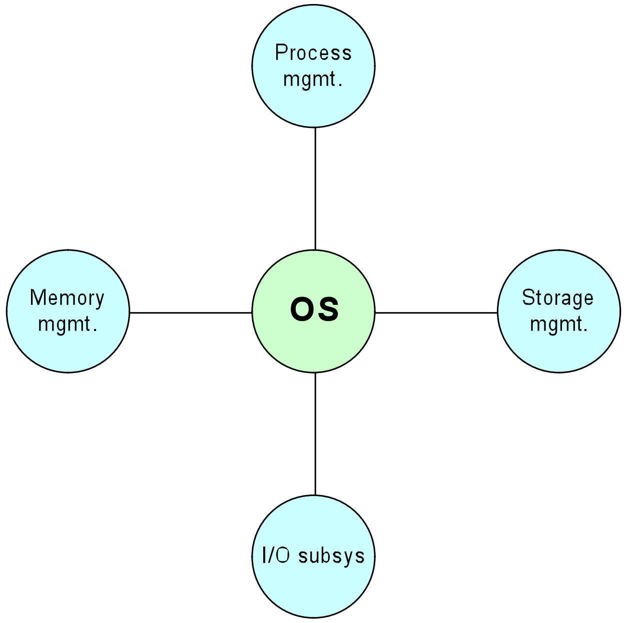
- Storage management
  - The primary storage is never big enough to accommodate all needs
  - A hierarchy of memory:
    - Size? Price (per MB)? Capacity? Bandwidth?
  - Move data in/out of hierarchy



OS Structure - I/O subsystem

- The I/O subsystem hides the oddities of individual I/O devices

- Instead, it provides a uniform interface (in Linux: a file)
  - The file is I/F to a device driver
  - The device driver knows how to operate the device



Operating systems - Real-Time OS’s

- Real-Time Operating Systems (RTOSs) are OSs intended for RT systems (!)

- ***Some key properties?***
  - Minimum interrupt latency
  - Minimum task switching latency
    - Includes known worst case latency (must be small)
  - Static task priorities

- The programmer (you!) is responsible for correct priority assignment