系統程式設計
Systems Programming

鄭卜壬教授
臺灣大學資訊工程系

Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

Contents

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2007
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

Chapter 0

- **Operating System Concepts Using UNIX as an Example**

UNIX is a kind of Operating Systems

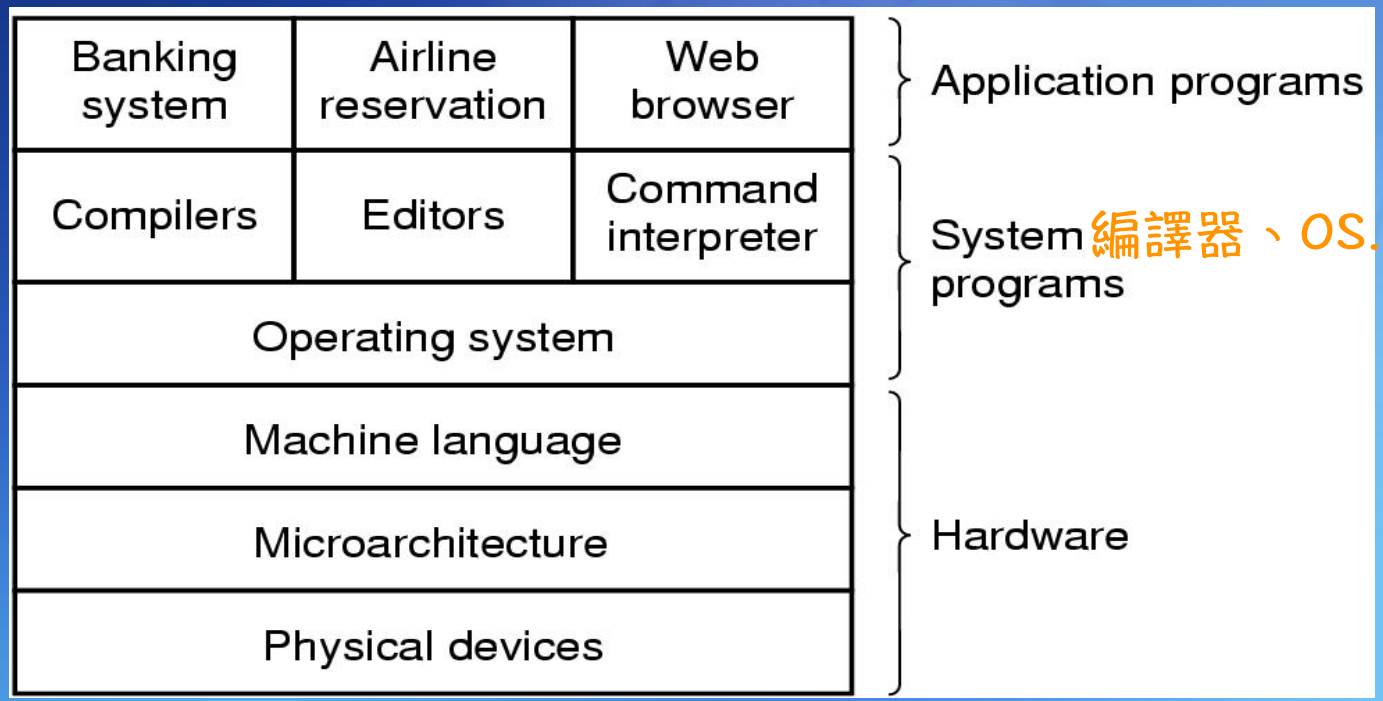
UNIX? 一種OS

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2007
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

A Computer System

- **A computer system consists of**
  - hardware
  - system programs
  - application programs

電腦系統包含？
硬體、系統程式、應用程式



What is an Operating System

垂直向上：是個VM，提供UI給使用者好用
向下：把底層雜事處理好

- **It is an extended machine (vertical)**
  - Presents user with a virtual machine, easier to use (establishes a user interface)
  - Hides the messy details which must be performed (executes and provides services **safely**)
- **It is a resource manager (horizontal)**
  - Resources: CPU, memory, I/O devices (disks, printers)
  - Each program gets time with the resource
  - Each program gets space on the resource
  - OS makes sure programs have a **fair** use

是個 資源管理者 (水平方向)
CPU, 記憶體、IO (磁碟、印表機都算)

What is an Operating System

- **It is an extended machine (vertical)**
  - Presents user with a virtual machine, easier to use (establishes a user interface)
  - Hides the messy details which must be performed (executes and provides services safely)
- **It is a resource manager (horizontal)**
  - Resources: CPU, memory, I/O devices (disks, printers)
  - Each program gets time with the resource
  - Each program gets space on the resource
  - OS makes sure programs have a fair use

UNIX Architecture (vertical viewpoint)

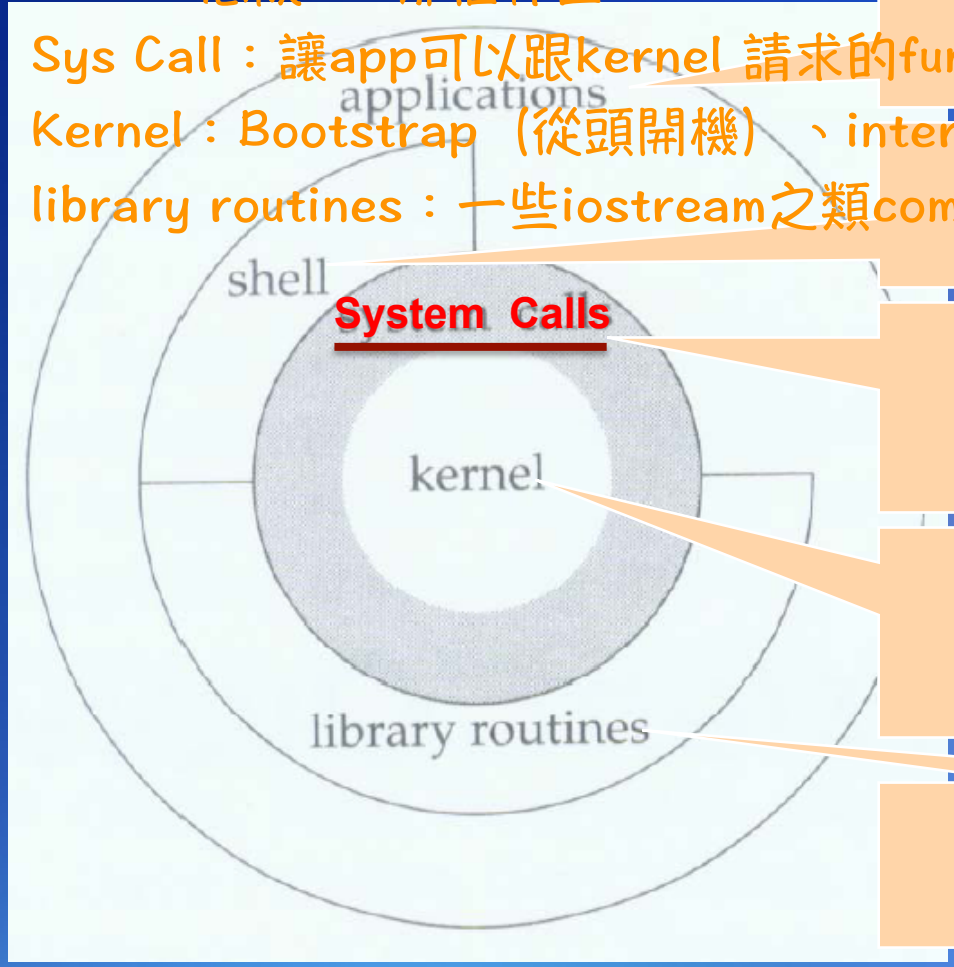
APP：user摸到的

Shell：想成cmd那個介面

Sys Call：讓app可以跟kernel 請求的function

Kernel：Bootstrap（從頭開機）、interrupt, excep, process…etc

library routines：一些iostream之類compile好的東西



Written by programmer
Compiled by programmer

Interactive interface
Users can issue commands (e.g., ls)

Application program can request service from kernel
Portable OS layer

Bootstrap, system initialization, interrupt and exception, process, memory &I/O management

Provided pre-compiled object codes
Defined in headers (e.g., stdio.h)

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2007
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

Interactive Interface to Unix

OpenSSH SSH client (remote login program)

Open SSH (用來遠端login的程式)

```
pjs-MacBook-Pro:~ pj$ ssh pjcheng@linux1.csie.ntu.edu.tw
pjcheng@linux1.csie.ntu.edu.tw's password:

#############################################################
#        Public Domain Workstation Lab (R217).              #
#############################################################
#   UNIX Login Service:                                     #
#     FreeBSD - bsd1                                        #
#     Linux   - linux1, linux2, linux3, ... linux20         #
#                                                           #
#   Office open time:                                       #
#     08:30 ~ 17:00, otherwise please use accesscards       #
#                                                           #
#   Contact information:                                    #
#     Web:    http://wslab.csie.ntu.edu.tw/                 #
#     E-Mail: ta217@csie.ntu.edu.tw                         #
#                                                           #
############################ Last Update: Dec   7 2014 ###

Last login: Tue Sep 15 22:30:37 2015 from 118.168.113.66
pjcheng@linux1:~>
```

shell prompt: where you type commands

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2008
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

```
pjcheng@linux1:~> cd SysProg
pjcheng@linux1:~/SysProg> ls
buffer.c  busywait.c  printstack.c  SSLServer.c  syscall.c  test_retaddr.c  thread_signal.c
pjcheng@linux1:~/SysProg> cat syscall.c
#include <unistd.h>
#include <sys/syscall.h>
#include <stdio.h>
#include <string.h>

int main()
{
        char * hello_with_syscall = "Hello World with syscall\n";
        char * hello_without_syscall = "Hello World without syscall\n";
        char * hello_with_printf = "Hello World with printf\n";
        write( 1, hello_without_syscall, strlen( hello_without_syscall ));
        syscall( SYS_write, 1, hello_with_syscall, strlen( hello_with_syscall ));
        printf( "%s", hello_with_printf );
}

pjcheng@linux1:~/SysProg> gcc -Wall syscall.c -o syscall
pjcheng@linux1:~/SysProg> ./syscall > outfile
pjcheng@linux1:~/SysProg> more outfile
Hello World without syscall
Hello World with syscall
Hello World with printf
pjcheng@linux1:~/SysProg> ls -al outfile
-rw-r--r-- 1 pjcheng users 77  9?? 12 11:22 outfile
pjcheng@linux1:~/SysProg> logout
Connection to linux1.csie.ntu.edu.tw closed.
```

- change the working directory
- list directory contents
- print on the standard output

System calls: write(), syscall()
Function call: printf() (C library)

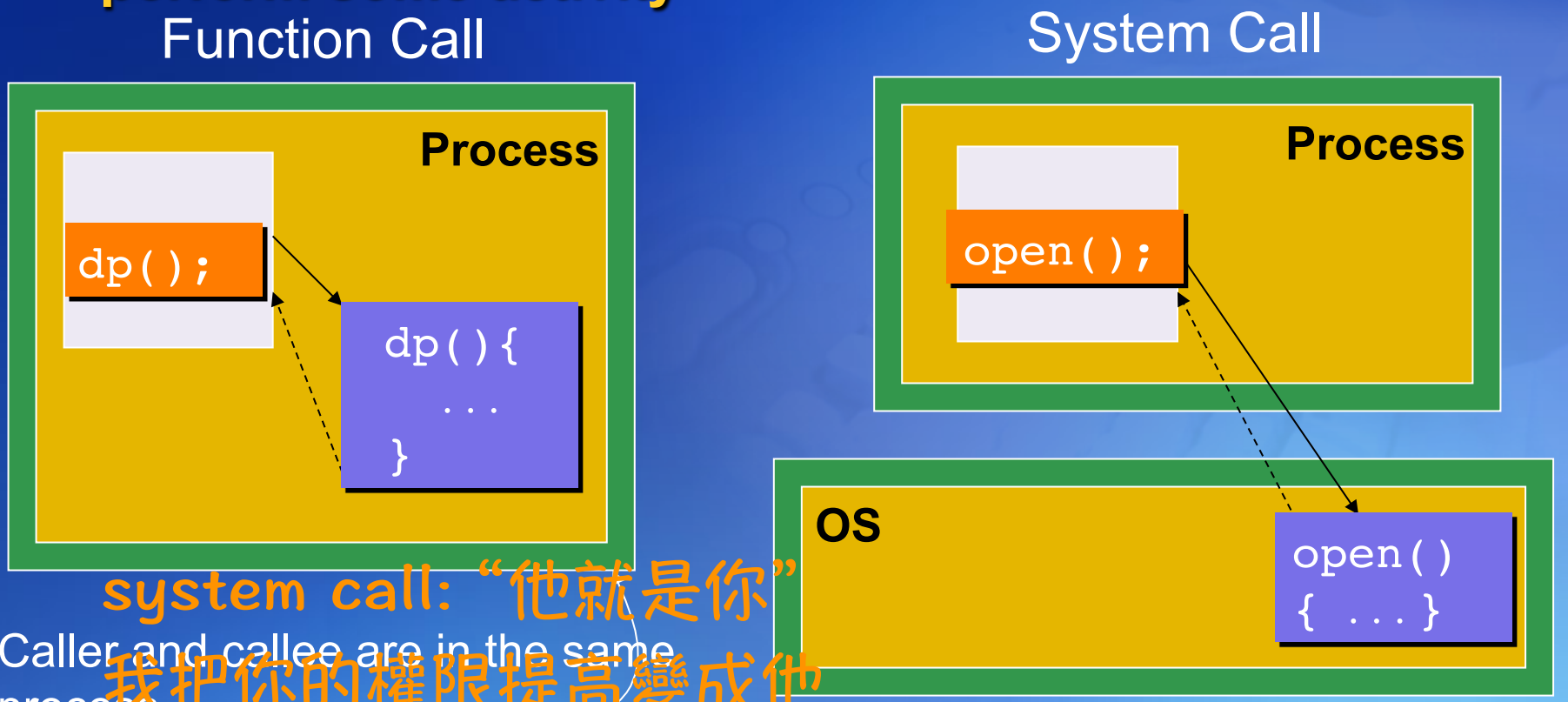
- compile with gnu C/C++ compiler
- run the program & redirect its standard output to a file
- paging through text one screenful at a time

man command

- Interface to online manuals

System Calls vs. Function Calls

- **System Call: a request to the operating system to perform some activity**



system call: "他就是你"

我把你的權限提高變成他

Caller and callee are in the same process
- Same user
- Same "domain of trust"

- OS is trusted; user is not.
- OS has super-privileges; user does not
- Must take measures to prevent abuse

User Mode vs. Kernel Mode

執行的模式有兩個

user mode

kernel mode

- **Modes of Execution (for protection)**
  - User mode vs. kernel mode
- **Most CPUs support at least two modes of execution: privileged (kernel-mode) and non-privileged (user-mode)**
- **User mode: a non-privileged mode in which processes are forbidden to access those portions of memory that have been allocated to the kernel or to other programs.**
- **When a user mode process wants to use a service that is provided by the kernel (e.g. a system call), the system must switch temporarily into kernel mode.**

User Space vs. Kernel Space

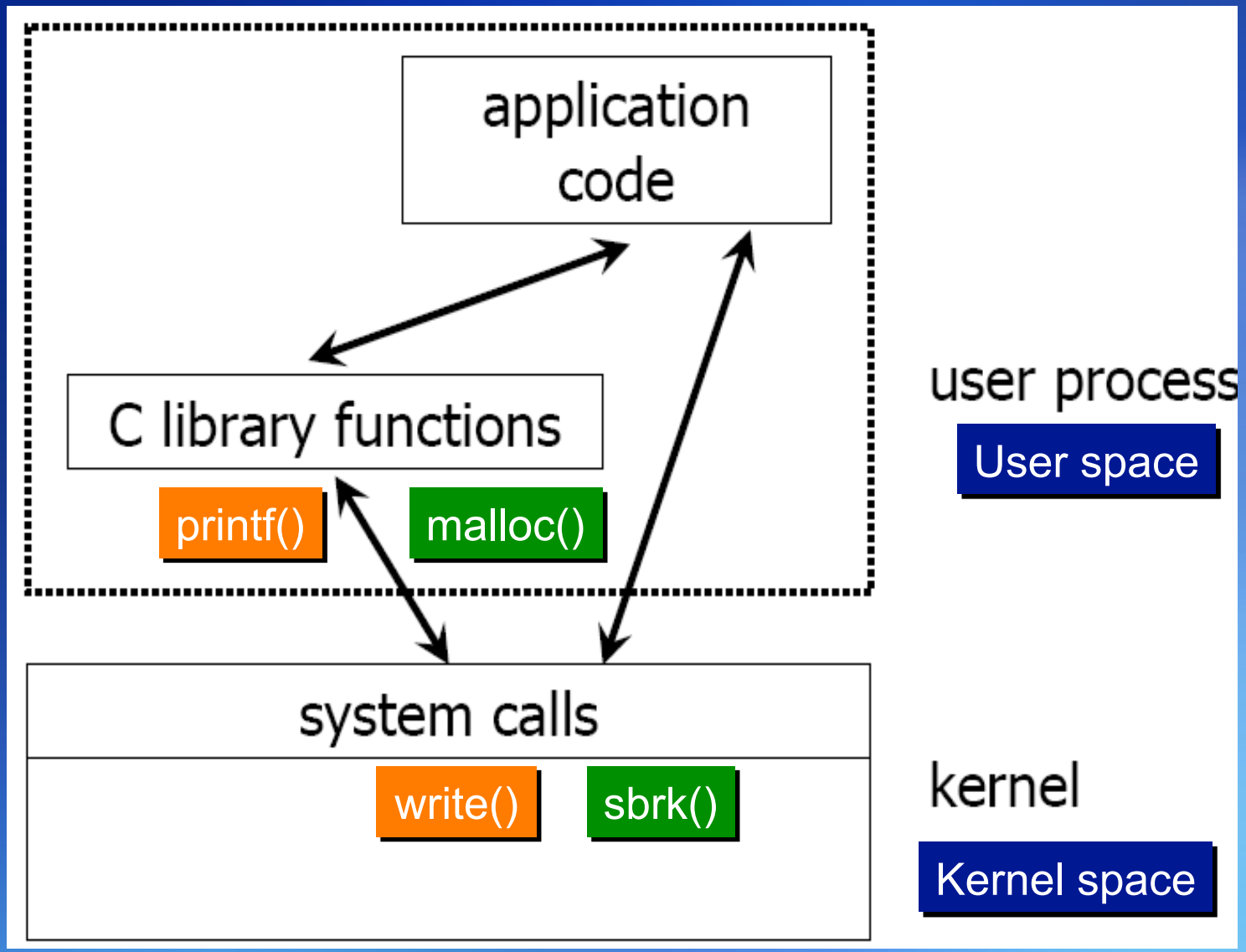
- **System memory is divided into two parts**
  - User space
    - a process executing in user mode is executing in user space
    - each user process is protected (isolated) from another (except for shared memory segments and mmapings, which will be discussed later)
  - Kernel space
    - a process executing in kernel mode is executing in kernel mode
- **Kernel space is the area wherein the kernel executes**

  user space is the area where a user program normally executes, *except when it performs a system call.*

當你在跑user mode的時候，
才會使用user space

當你在跑kernel mode的時候，
才會使用kernel space

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2007
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University



Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2007
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

Example 1

```
void main()
{
        int i, sum=0;

        for (i=1; i<=100000000; i++)
                sum += i;
}
```

Shell command: time –p execfile

(time: run execfile & summarize time usage)

A) real 0.43, user 0.42, sys 0.00 (in seconds) ? or

B) real 2.99, user 0.04, sys 1.32 (in seconds) ?

real: s

user

sys:

```
void main()
{
        int i;
        char *str = "Hello World\n";

        for (i=1; i<1000000; i++)
                write( 1, str, strlen( str ));
}
```

Example 2

```
void main()     // busy wait for 5 seconds
{
        time_t start_time;

        start_time = time( NULL );
        while (1) {
                if ( time( NULL ) > start_time + 5 ) break;
        }
}
```

real 5.60, user 5.50, sys 0.00 (bad)

Comparison: time –p sleep 5

real 5.00, user 0.00, sys 0.00 (good)

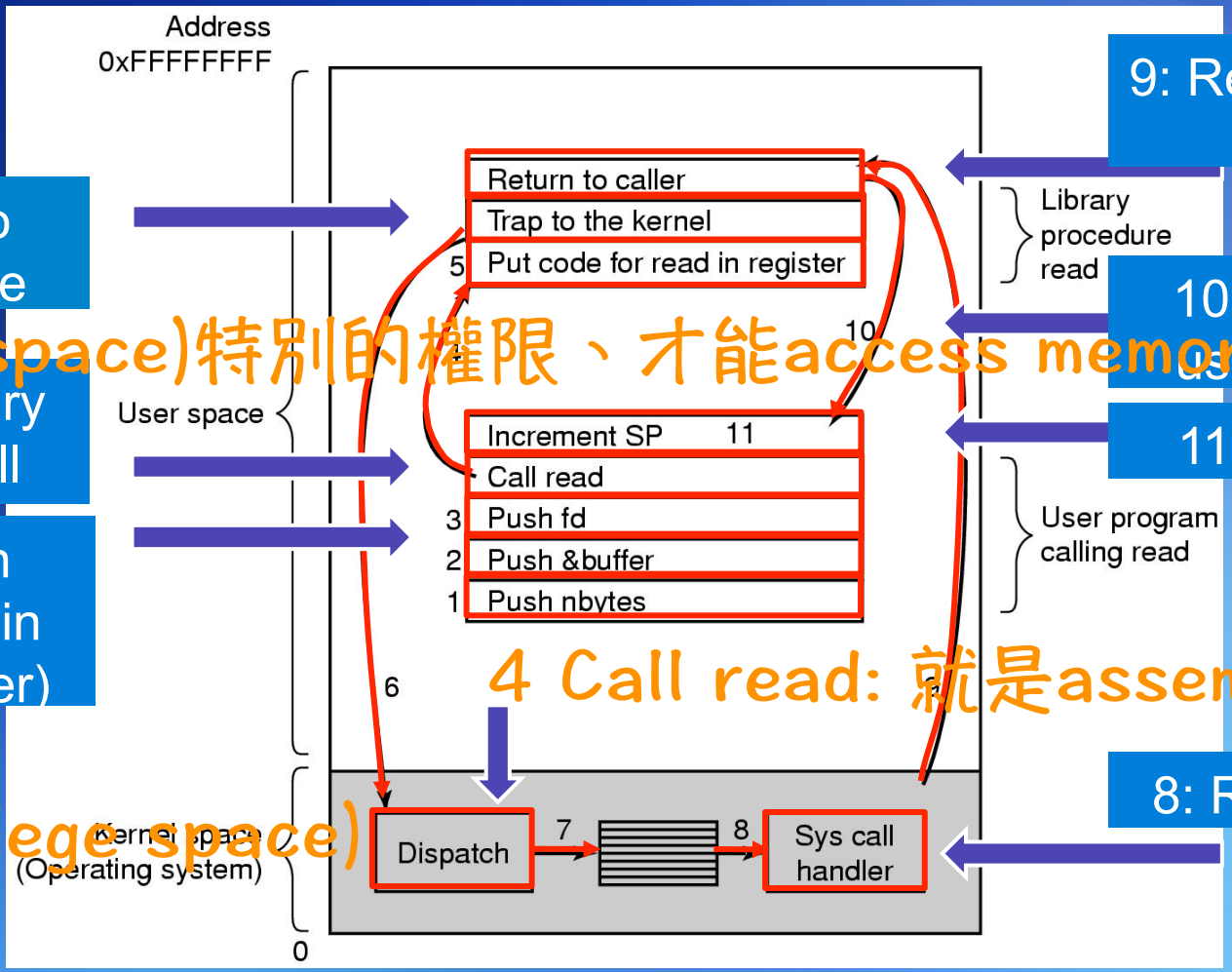
(sleep: delay for a specified amount of time)

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2007
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

Steps for Making a System Call

Making a system call is expensive (Reading: concept of system call)

Example: **read (fd, buffer, nbytes)**



9: Return to user mode

10: Return to user program

11: Clean up

8: Run handler

6: Switch to kernel mode

(privilege space)特別的權限、才能access memory

4 – 5: C library function call

1 – 3: Push parameter (in reverse order)

4 Call read: 就是assembly的jump

(non-privilege space)

See also: long syscall(long number, …)

What is an Operating System

memory != RAM memory

而是virtual memory

每一個process都有一個stack

- It is an extended machine (vertical)
  - Presents user with a virtual machine, easier to use (establishes a user interface)
  - Hides the messy details which must be performed (executes and provides services safely)
- It is a resource manager (horizontal)
  - Resources: CPU, memory, I/O devices (disks, printers)
  - Each program gets time with the resource
  - Each program gets space on the resource
  - OS makes sure programs have a fair use

Resource Manager (horizontal viewpoint)

Services for Application Programs:

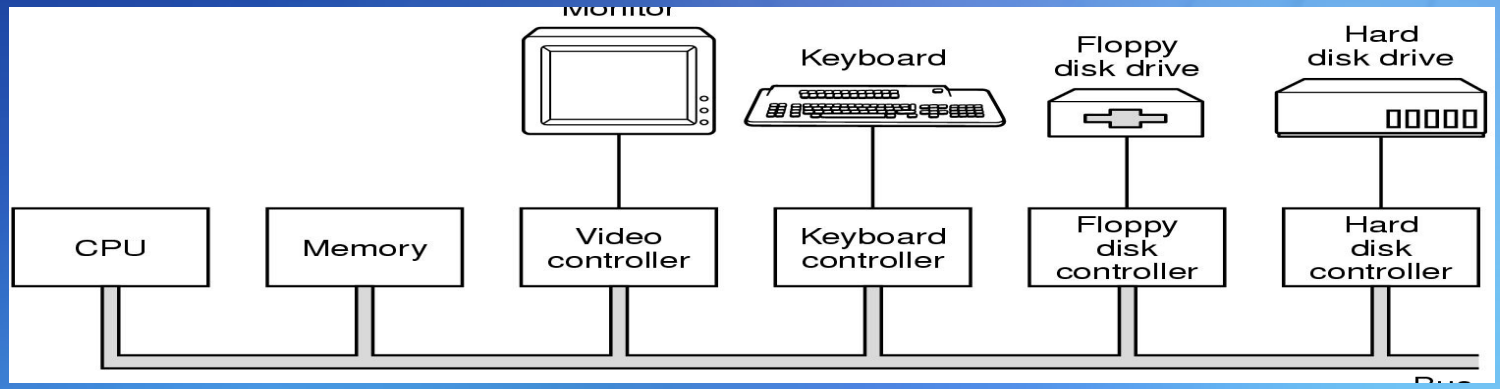
User Identification, Process Management,
Memory Management, File/Directory Management,
Inter-process Communication, Signal, I/O Management
(e.g., Terminal, Network, etc), …

intra : process之間

inter : process內

OS Kernel:

CPU Scheduling, Virtual Memory, File System,
Protection, Security, Synchronization, I/O Control, …



Example of hardware components for a PC

Services for Application Programs

Chapter 1

- User Identification
- Process Management
- Memory Management
- File/Directory Management

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2008
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

Services for Application Programs

- **User Identification**
- Process Management
- Memory Management
- File/Directory Management

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2008
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

User Identification in UNIX

- **Logging In** **(See Ch1.3, 6.2, 6.3)**
  - /etc/passwd – local machine or NIS DB
    - root:x:0:1:Super-User:/root:/bin/tcsh
    - Login-name, encrypted passwd, numeric user-ID, numeric group ID, comment, home dir, shell program
  - /etc/shadow – with “x” indicated for passwd

user帳號/加密後的密碼/user-ID/group-id

user-ID:0代表unix內的superuser

- **Related shell command:**

  passwd (i.e., change user password)

Shell command: cat /etc/passwd

system administrator

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
```

uid = userid

root = superuser

nobody = 無特殊權限

>su //想要換成superuser的身份

>su AdrianHsu //換成AdrianHsu的身份

Password Encryption and Salt

- *crypt(3)* is designed to make a key search computationally expensive
- /bin/passwd selects a salt based on the time of day
- Salt is converted into a two character string and stored in the encrypted password file

crypt(“apple”, “am”) = “amADBMNoVAZpc”
crypt(“water”, “pm”) = “pmRarHxhhU34U”

- Salt makes it harder for an attacker to build a reverse dictionary and allows users to use the same password on different computers

crypt(“x”, “y”)
字典攻擊法(把用字典一個個試)
salty bit: 就是y, 兩個char, 避免一樣的hash
保證hash後的前2個char = salty bit

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2007
Department of Computer Science and Information Engineering
Graduate Institute of Multimedia and Networking, National Taiwan University

Shell command: man 3 crypt

man: an interface to the on-line reference manuals

```
NAME
       crypt, crypt_r - password and data encryption

SYNOPSIS
       #define _XOPEN_SOURCE       /* See feature_test_macros(7) */
       #include <unistd.h>

       char *crypt(const char *key, const char *salt);

       #define _GNU_SOURCE         /* See feature_test_macros(7) */
       #include <crypt.h>

       char *crypt_r(const char *key, const char *salt,
                     struct crypt_data *data);

       Link with -lcrypt.

DESCRIPTION
       crypt() is the password encryption function.  It is based on the Data Encryption Standard algorithm with variations intended
       of a key search.

       key is a user's typed password.

       salt is a two-character string chosen from the set [a?VzA?VZ0?V9./].  This string is used to perturb the algorithm in one of

       By taking the lowest 7 bits of each of the first eight characters of the key, a 56-bit key is obtained.  This 56-bit key is u
       sisting  of all zeros).  The returned value points to the encrypted password, a series of 13 printable ASCII characters (the
       points to static data whose content is overwritten by each call.
```

Services for Application Programs

- User Identification
- **Process Management**
- Memory Management
- File/Directory Management

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2008
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

Programs and Processes

- Program
  - An executable file residing in a disk file
- Process
  - An executing instance of a program
  - Unique process ID
  - Related shell commands: ps, top

ps - report a snapshot of the current processes

top – display processes

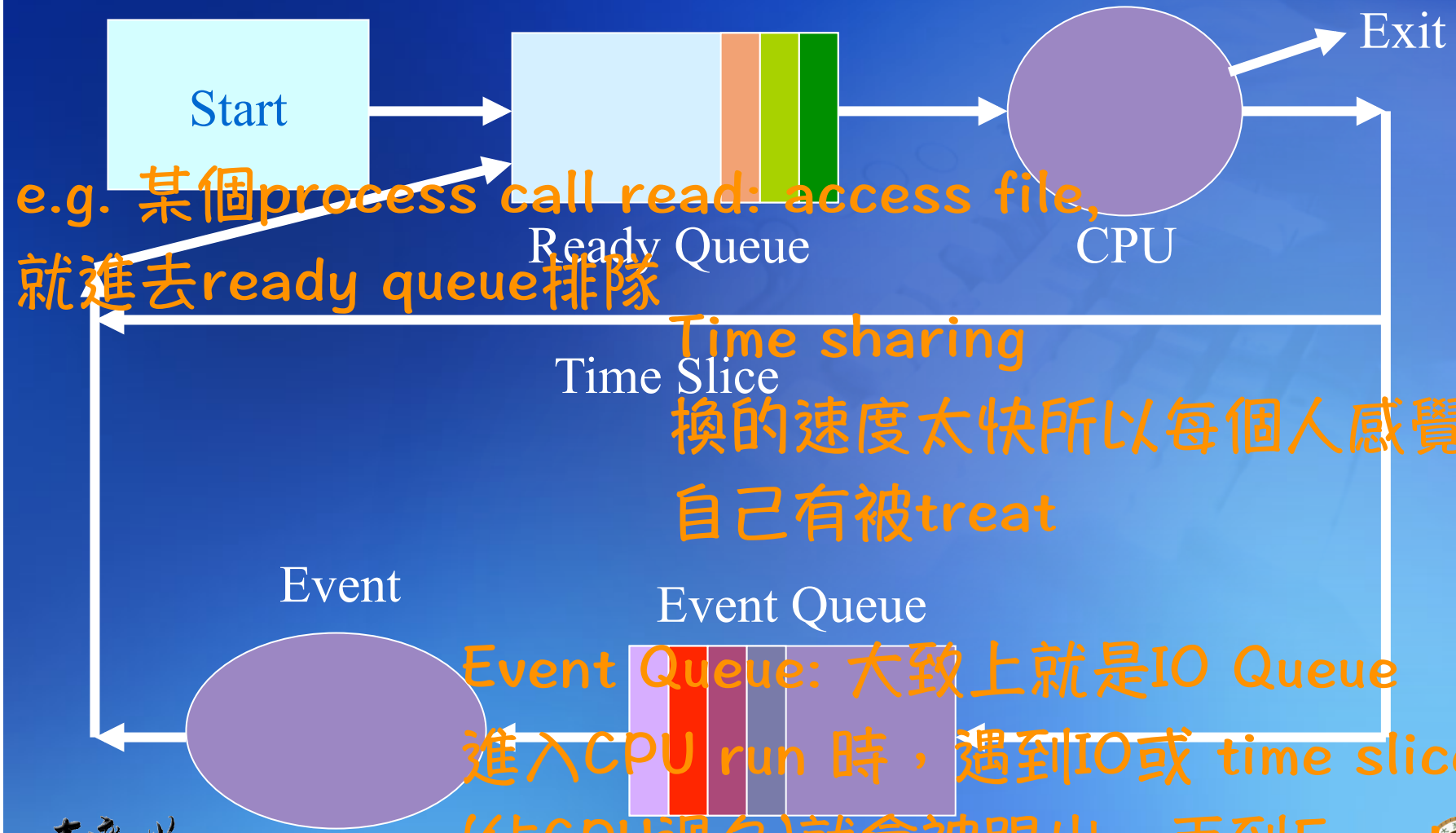
CPU Scheduling

- **Deciding which process should occupy the resource (CPU, disk, etc)**

- **User can change a process priority**
  - Related shell command: nice

Example of Multitasking

Time sharing: CPU's time is shared among multiple tasks simultaneously

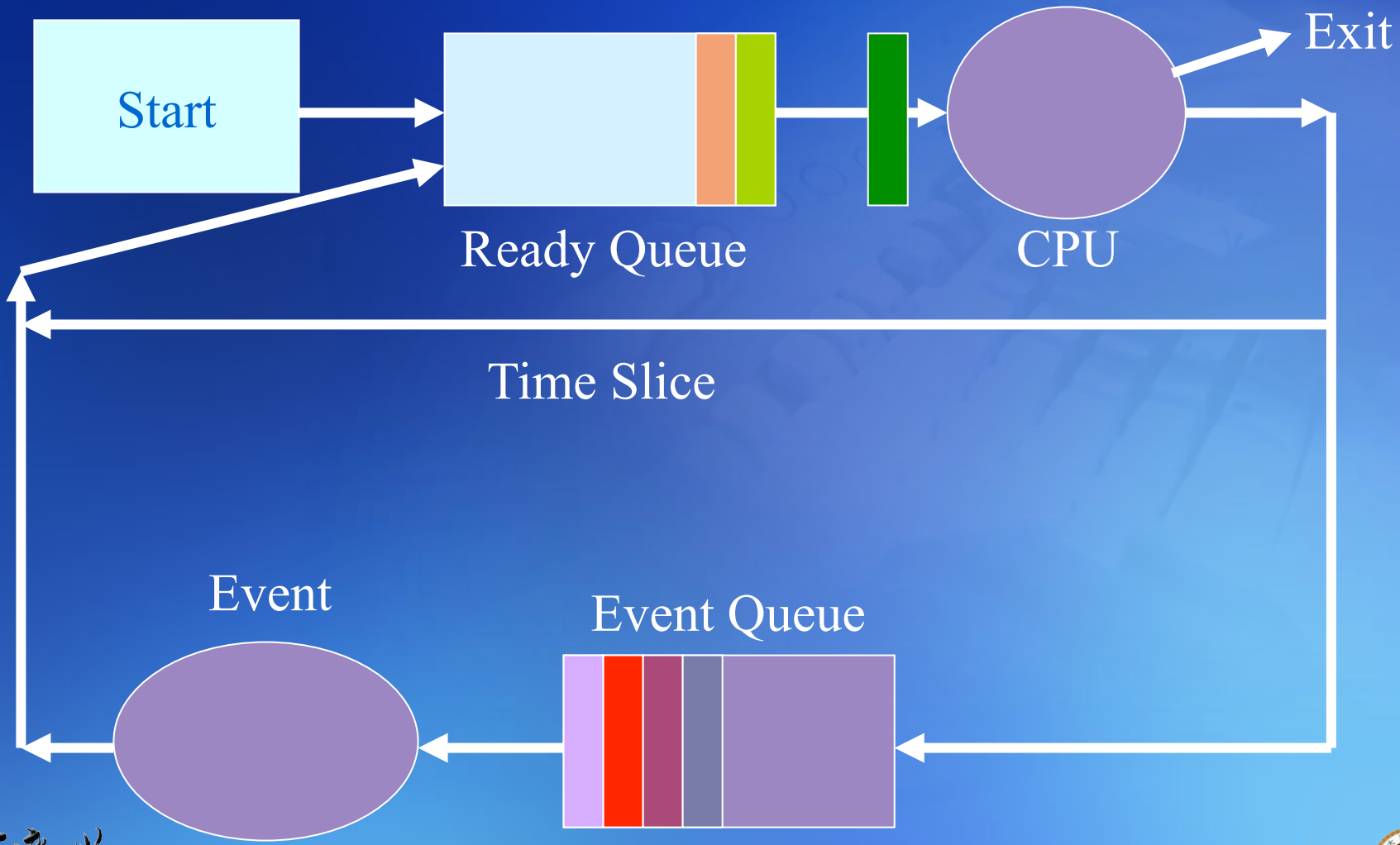


e.g. 某個process call read: access file,
就進去ready queue排隊

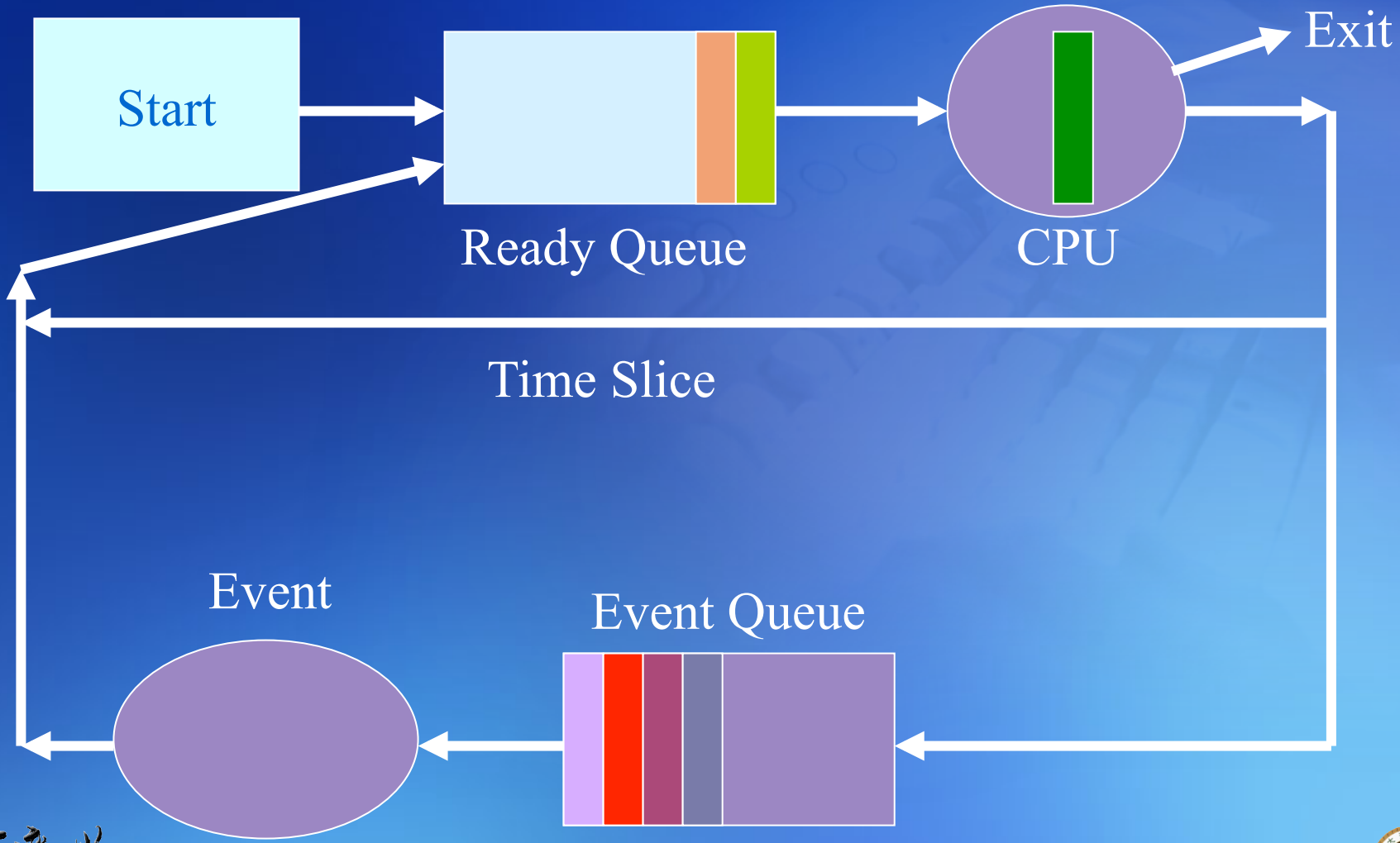
Time sharing
換的速度太快所以每個人感覺都覺得
自己有被treat

Event Queue: 大致上就是IO Queue
進入CPU run 時，遇到IO或 time slice 超過，
(佔CPU過久)就會被踢出、丟到Event Queue

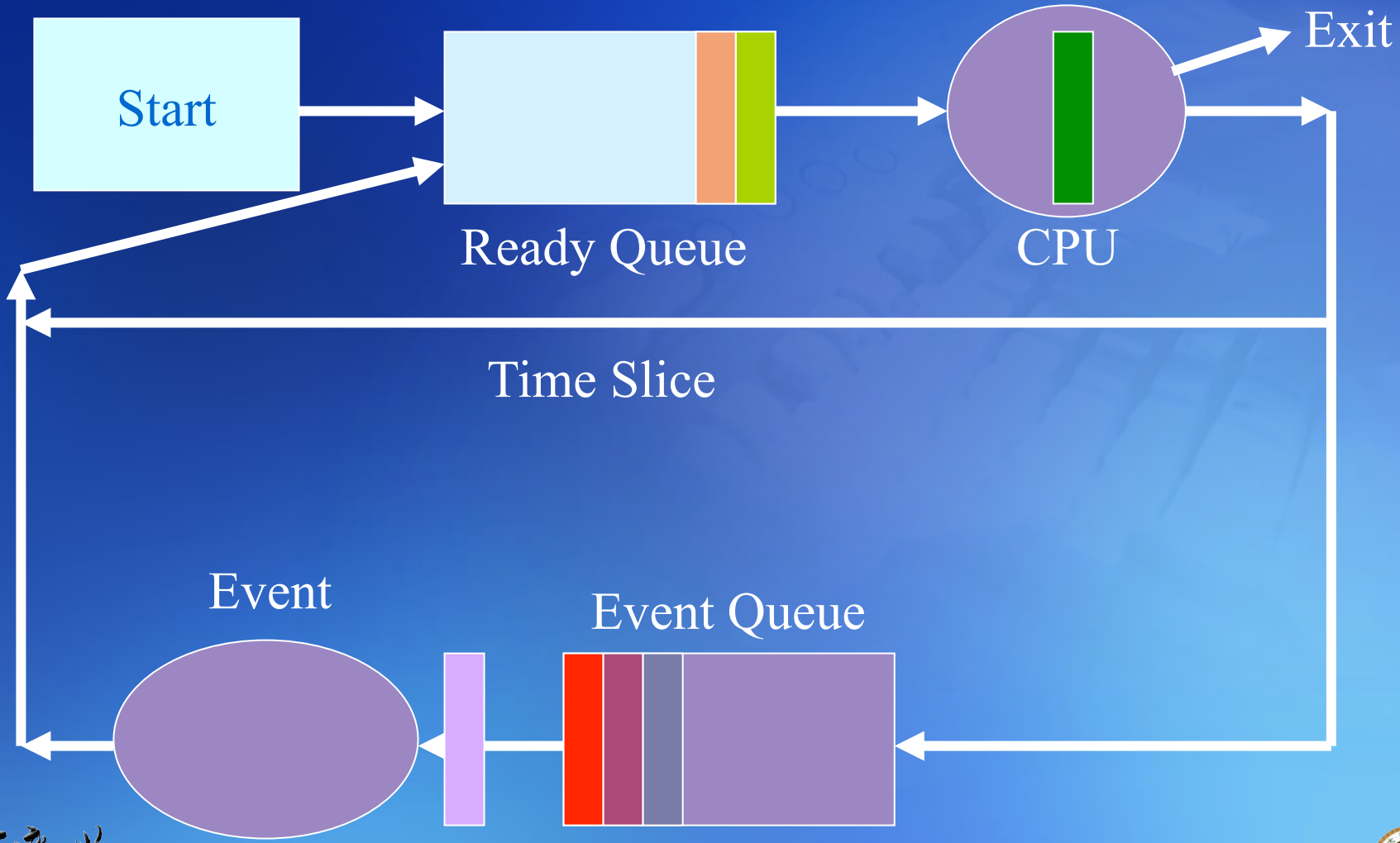
Example of Multitasking



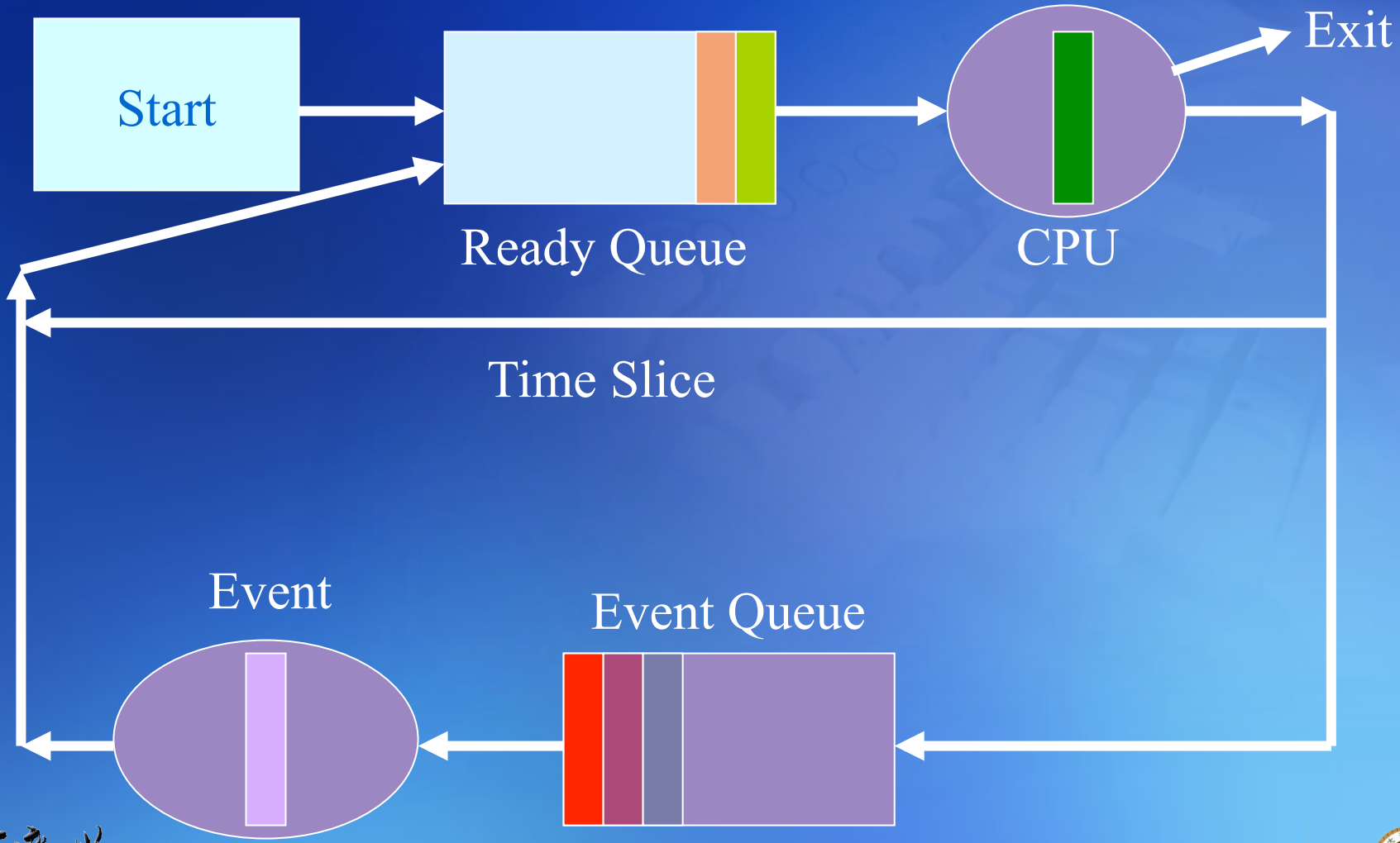
Example of Multitasking



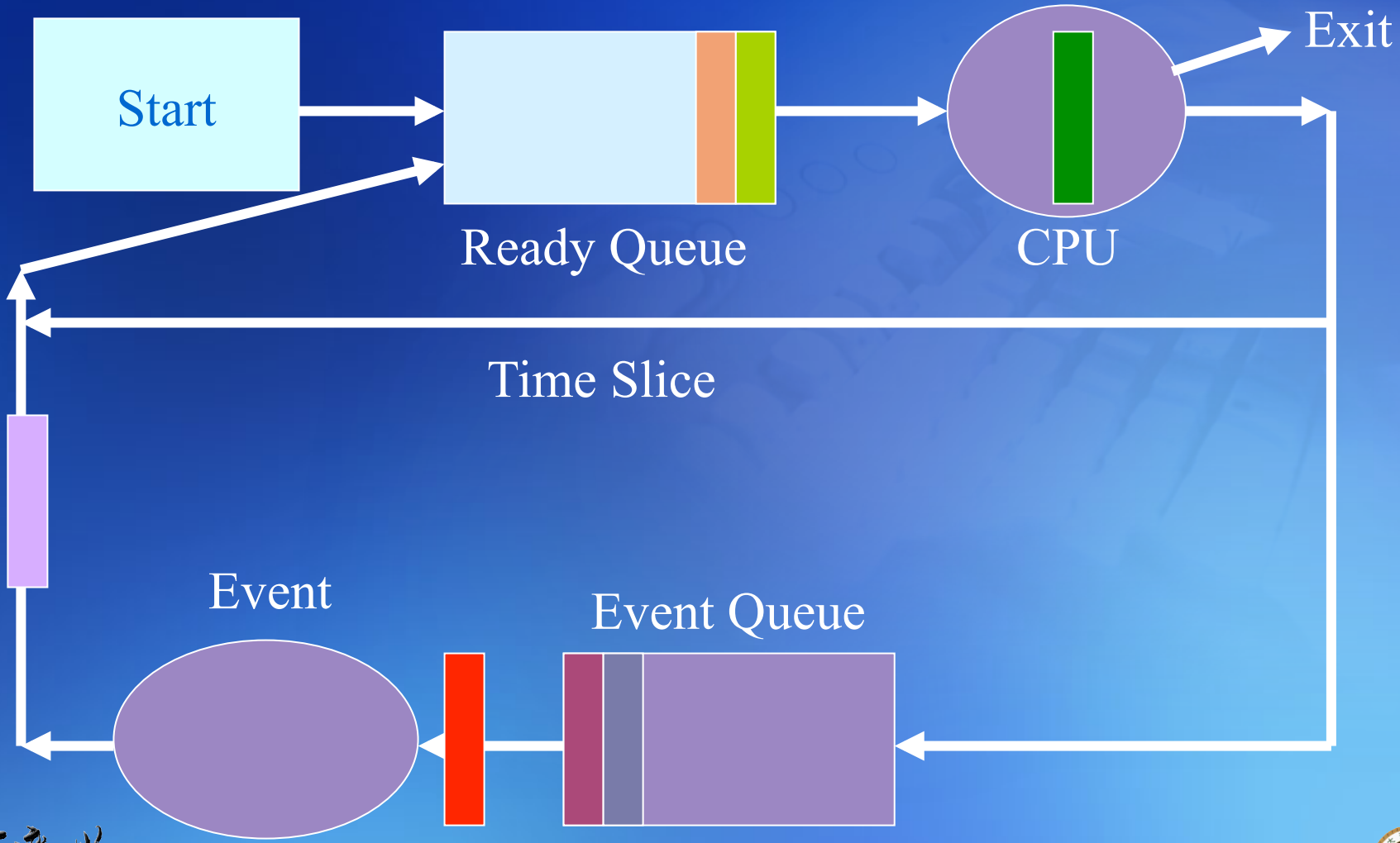
Example of Multitasking



Example of Multitasking

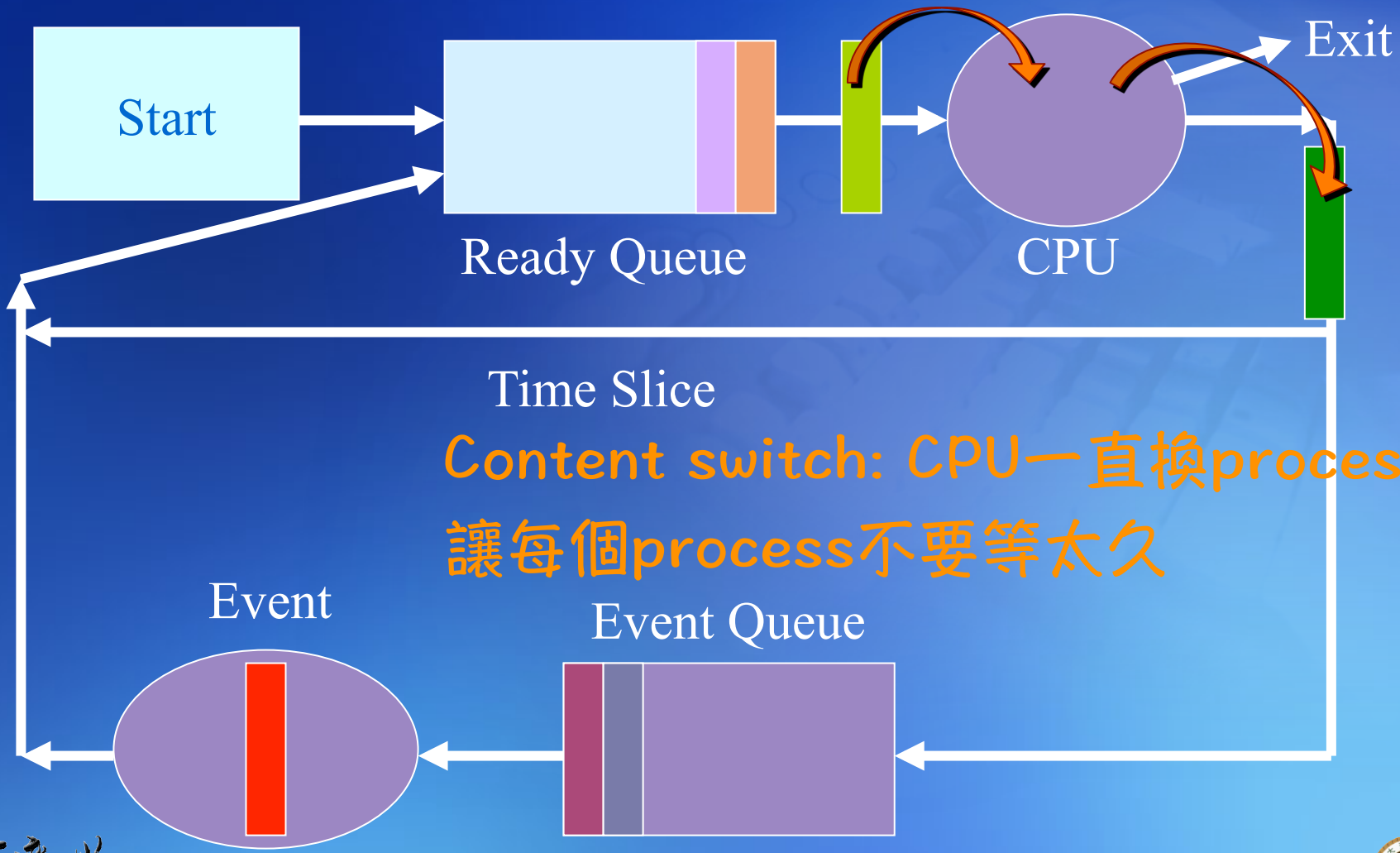


Example of Multitasking

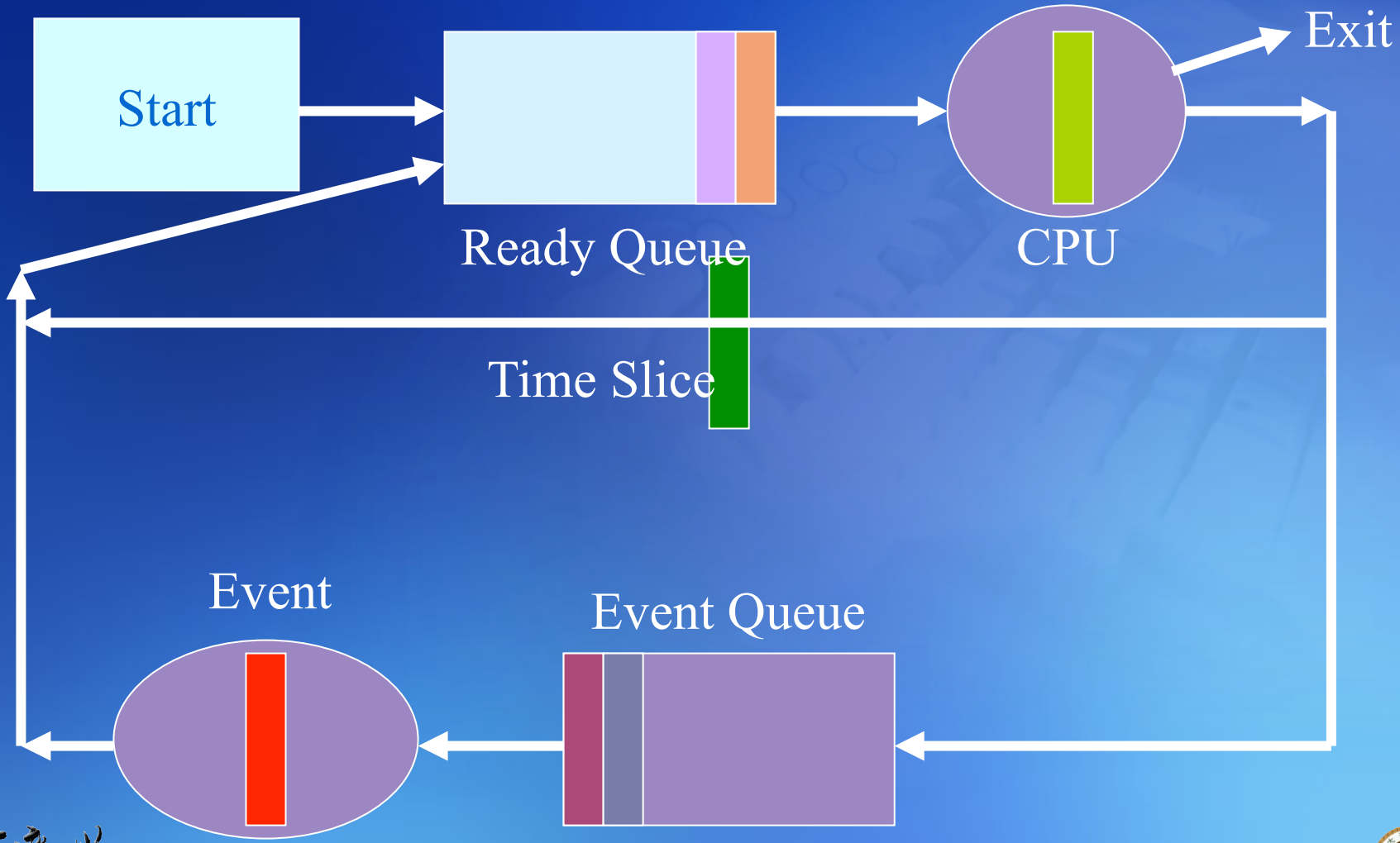


Example of Multitasking

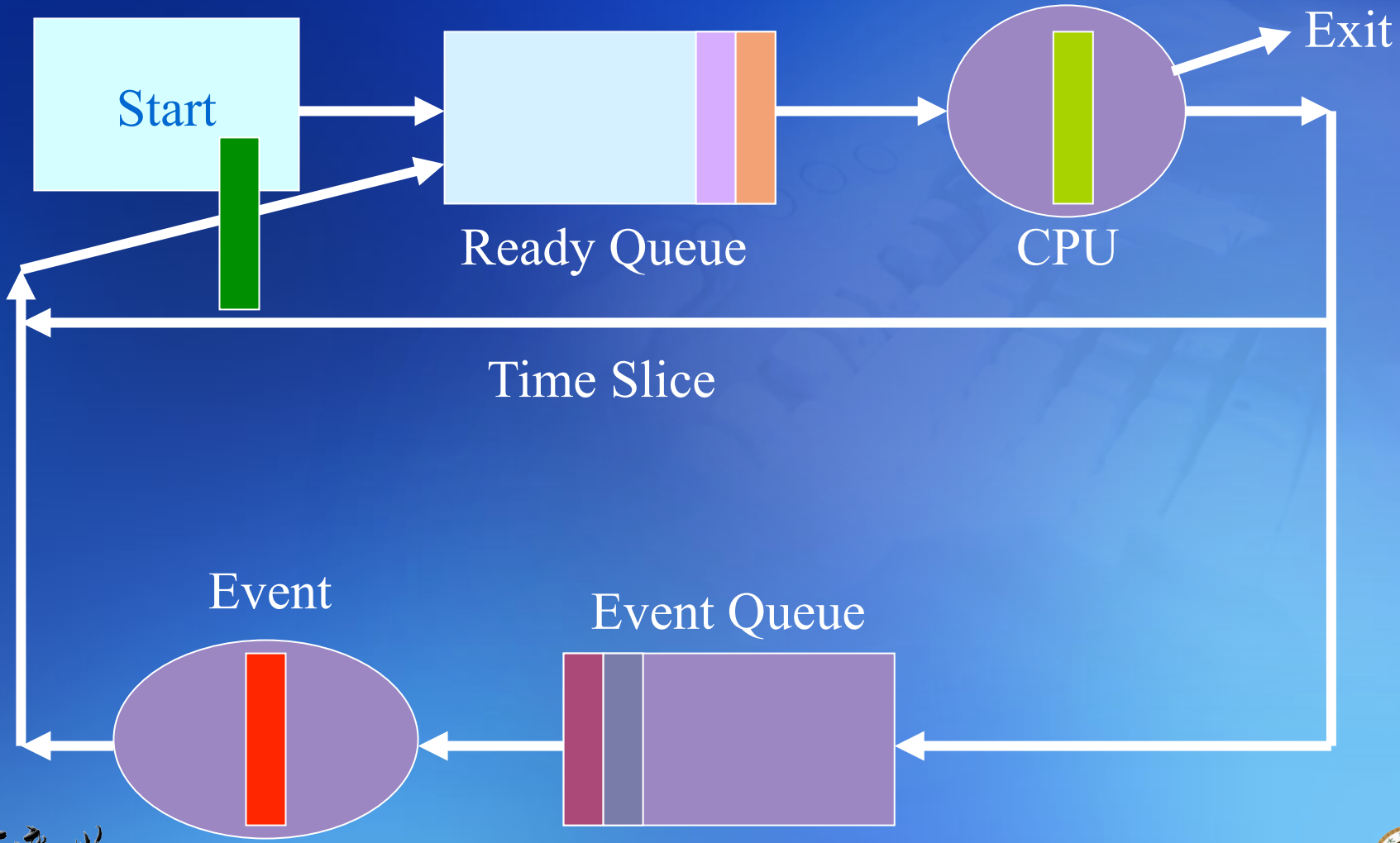
Context switch: process of storing and restoring execution context of a process



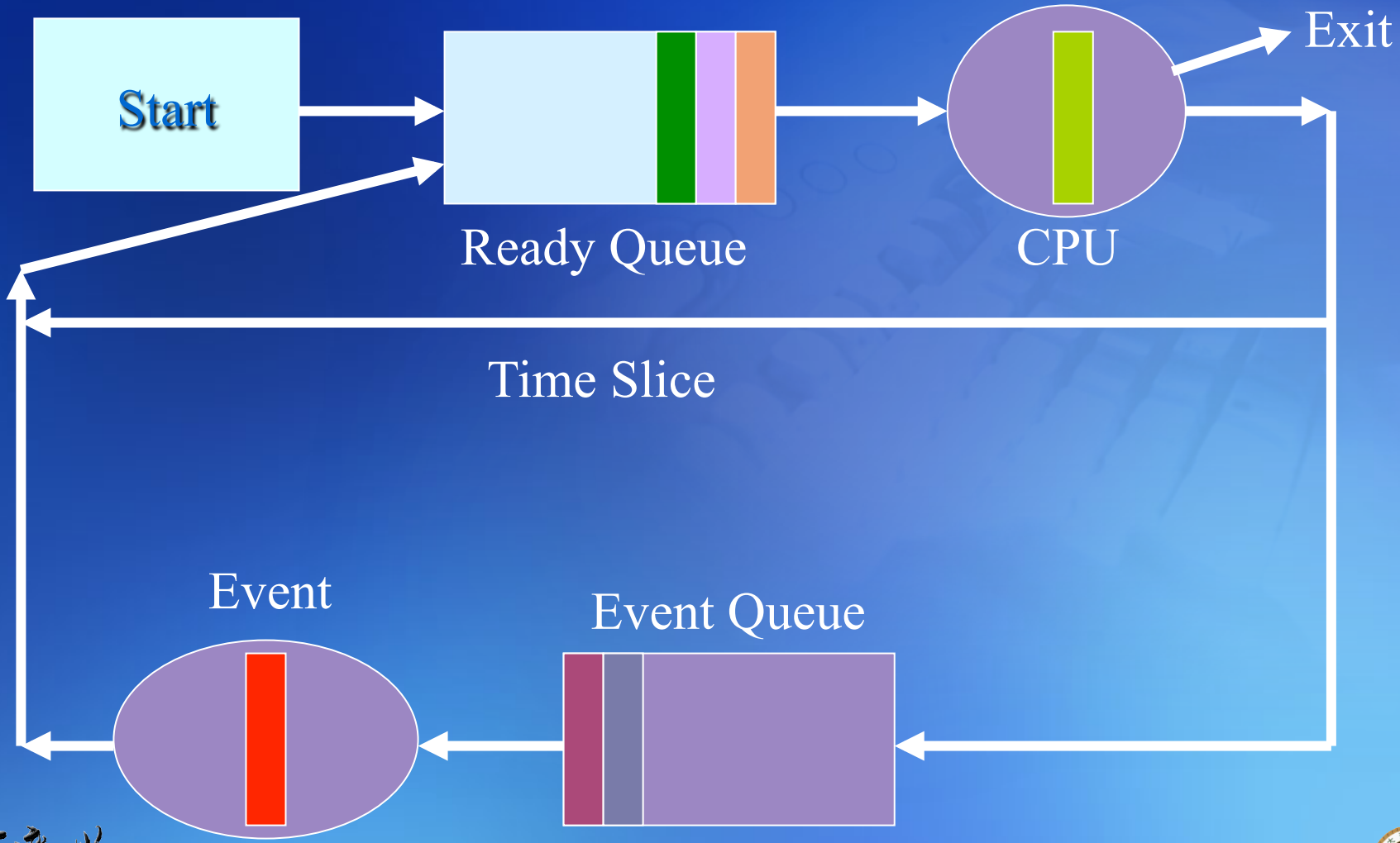
Example of Multitasking



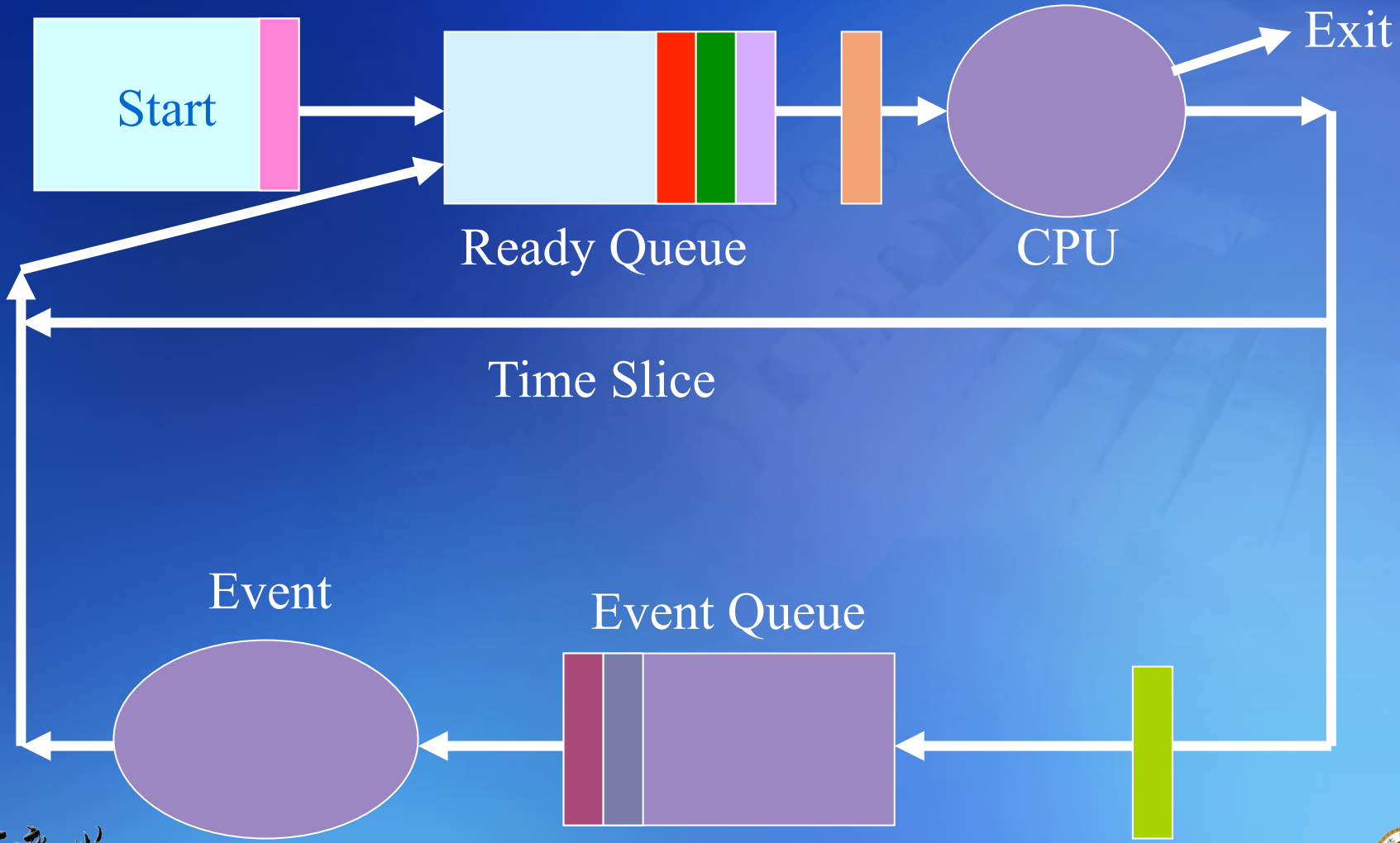
Example of Multitasking



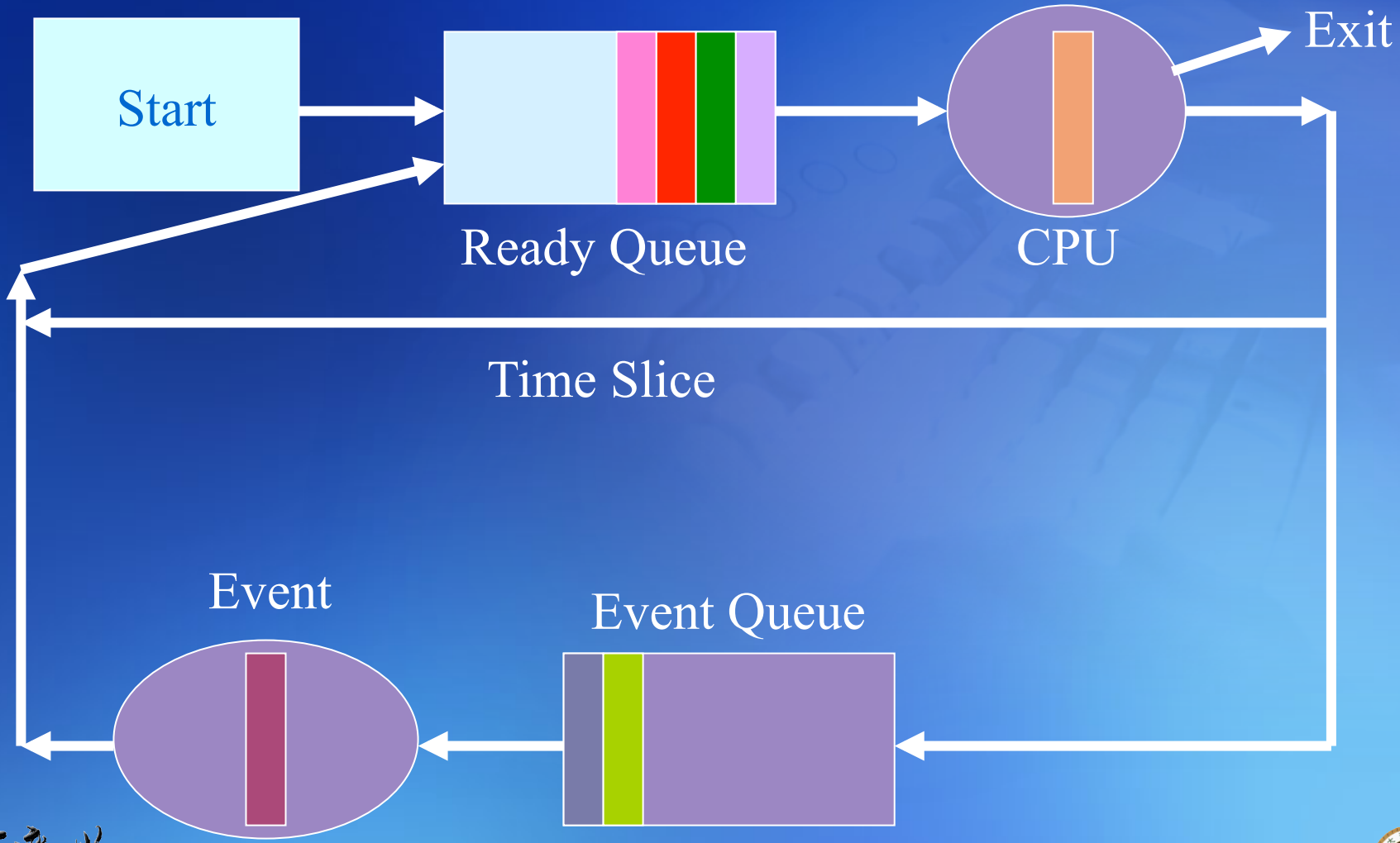
Example of Multitasking



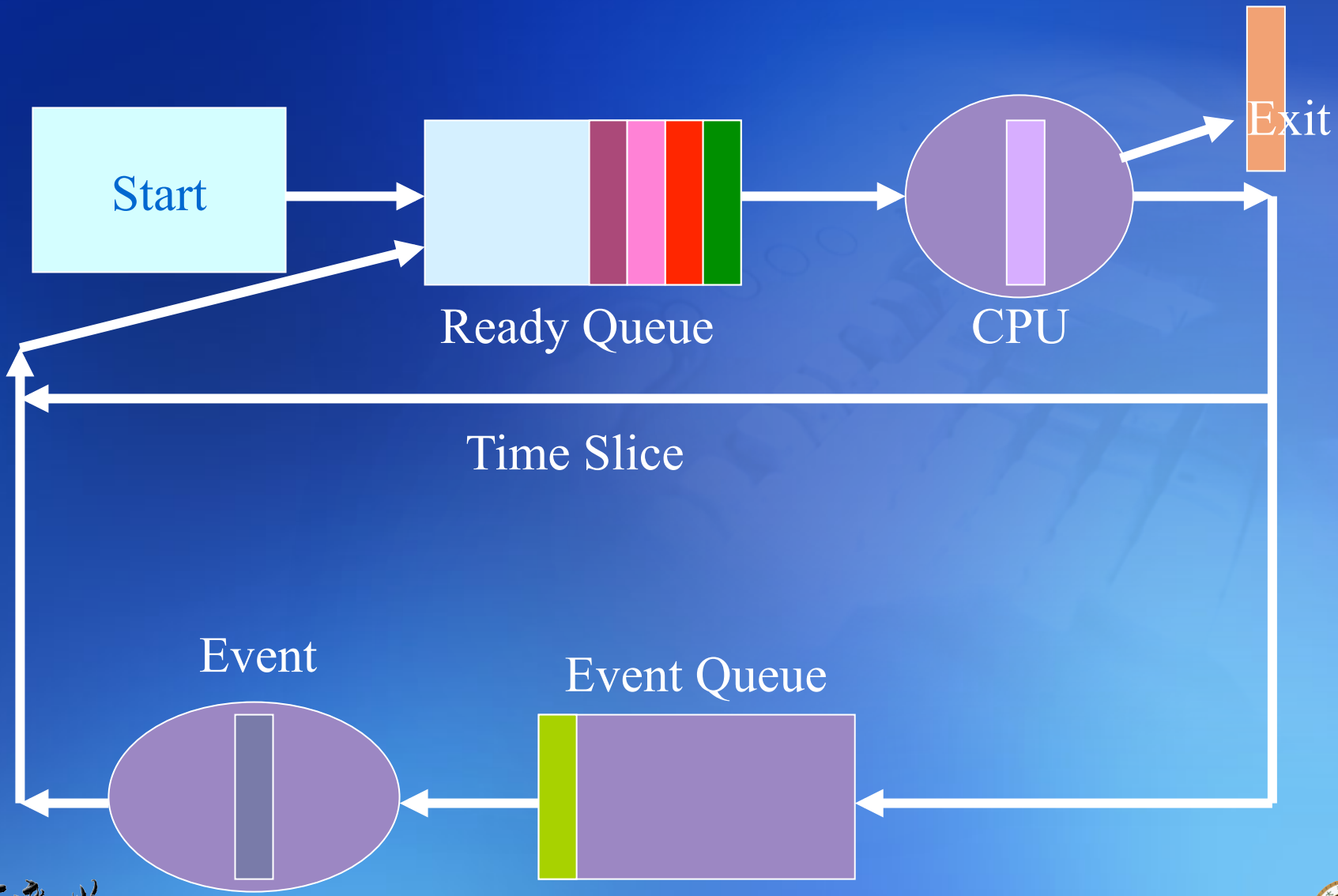
Example of Multitasking



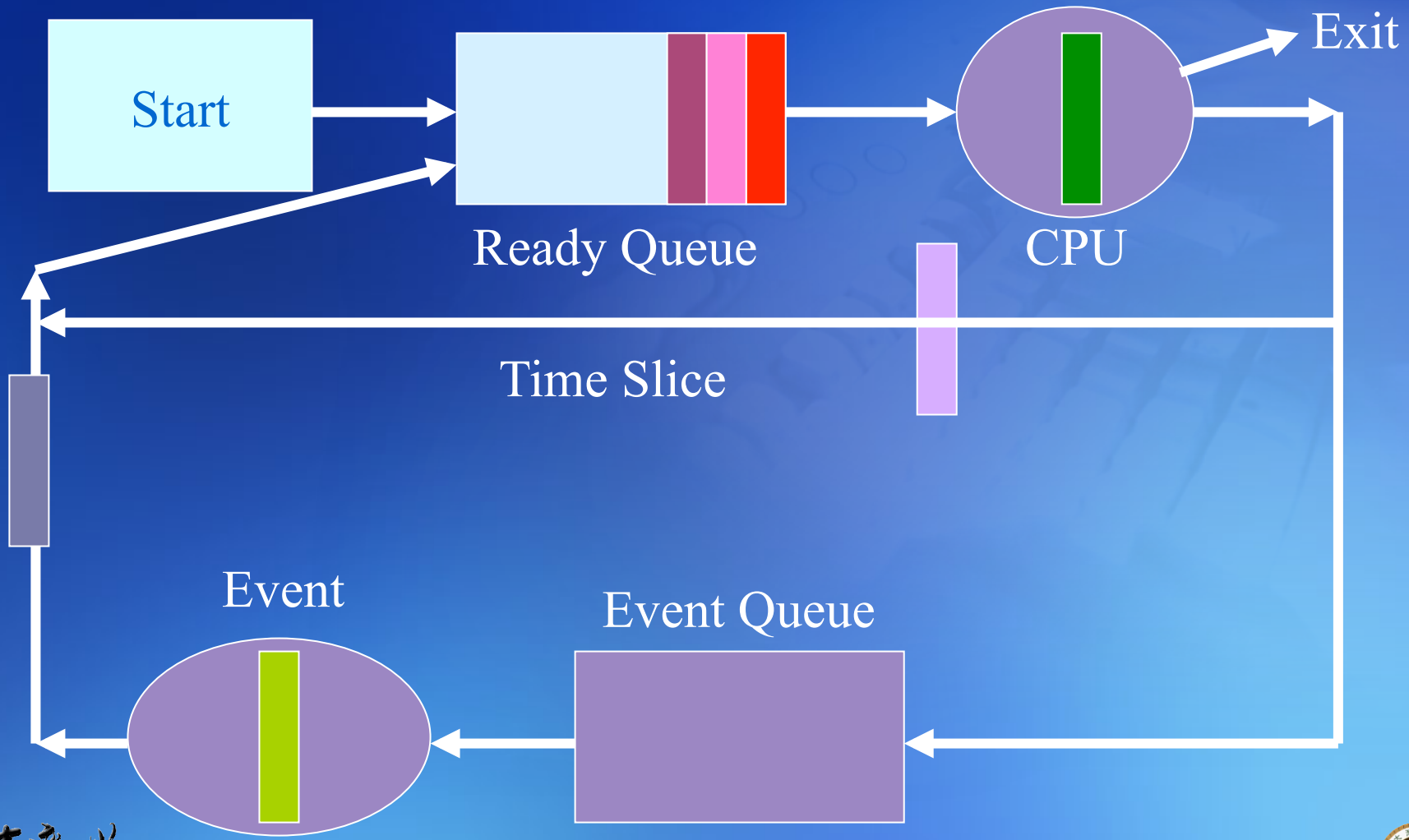
Example of Multitasking



Example of Multitasking



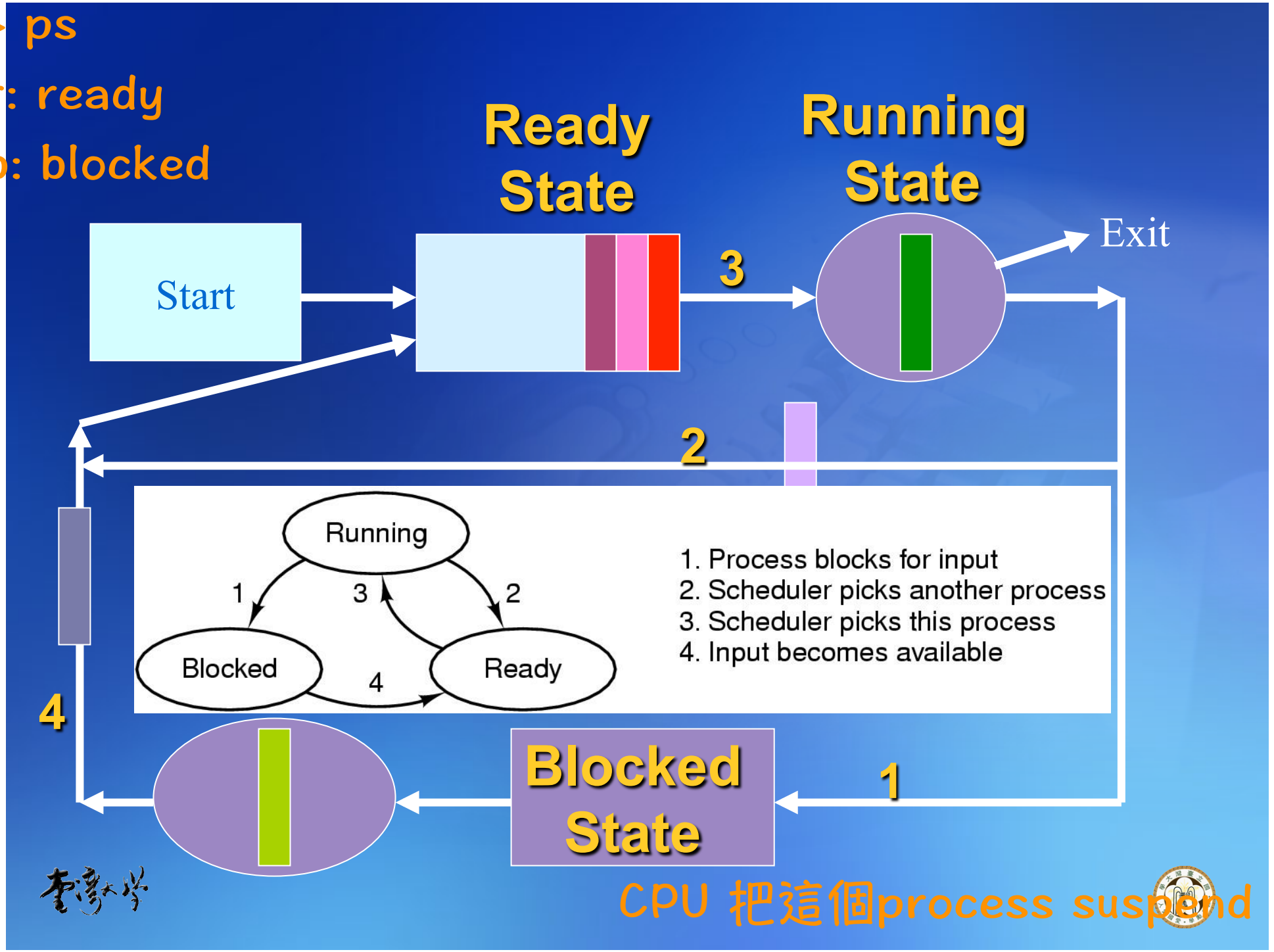
Example of Multitasking



> ps

r: ready

b: blocked



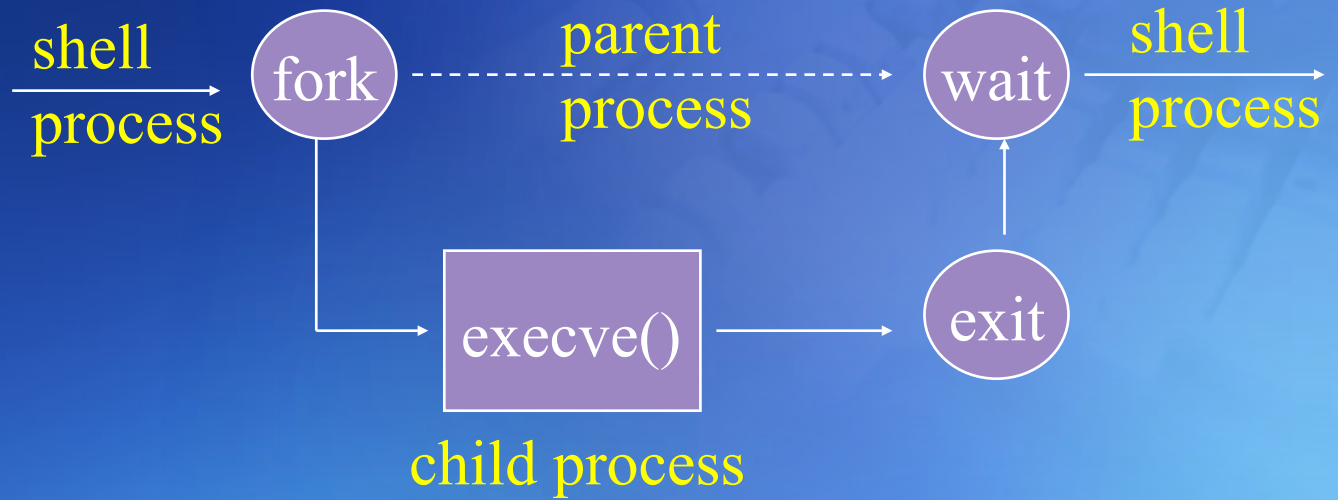
CPU 把這個process suspend

UNIX Process

fork 一大堆process, 把所有id領光，
其他人就不能用了 (雖然CPU這時很閒
打ls、就會跑下面這個流程圖

- **Shell**
  - Command interpreters

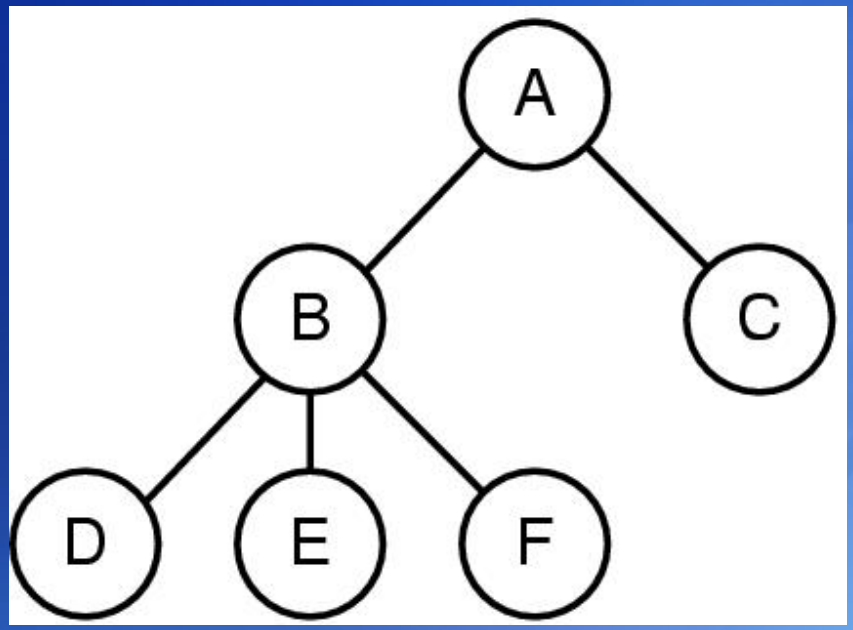
    e.g., ls, pwd



(Will be explained in details later; Ch8)

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2007
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

A Process Tree



- A created two child processes, B and C
- B created three child processes, D, E, and F

Common UNIX Shells

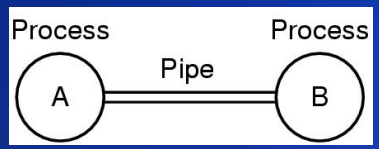
- **Shells**
  - Bourne shell, /bin/sh
    - Steve Bourne at Bell Labs
  - C shell, /bin/csh
    - Bill Jay at Berkeley
      - Command-line editing, history, job-control, etc
  - KornShell, /bin/ksh
    - David Korn (successor of Bourne shell)
    - Command-line editing, job-control, etc
  - Related shell command: chsh (i.e.,change shell)

可以換shell (利用chsh)

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2007
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

Inter-process Communication

- **Pipe**



  - A way to send the output of one command to the input of another
- **Filter**
  - A program that takes input and transforms it in some way
  - `wc` - gives a count of words/lines/chars
  - `grep` - searches for lines with a given string
  - `more`
  - `sort` - sorts lines alphabetically or numerically

生出(fork) A, B兩個小孩process，
並用pipe把某個output當成另一邊的input
e.g. A的output吐出來，給B當作input

Examples of Filter & Pipe

- `ls -la | more` more:
- `cat file | wc` wc:
- `man ksh | grep "history"`
- `ls -l | grep "bowman" | wc`
- `who | sort > current_users` sort > current_users

(Try these commands yourself)

UNIX philosophy:

- Write programs that do one thing and do it well
- Write programs that work together
- Write programs that handle text streams, because that is the universal interface

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2007
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

Some System Calls For Process Management

Process management	
Call	**Description**
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

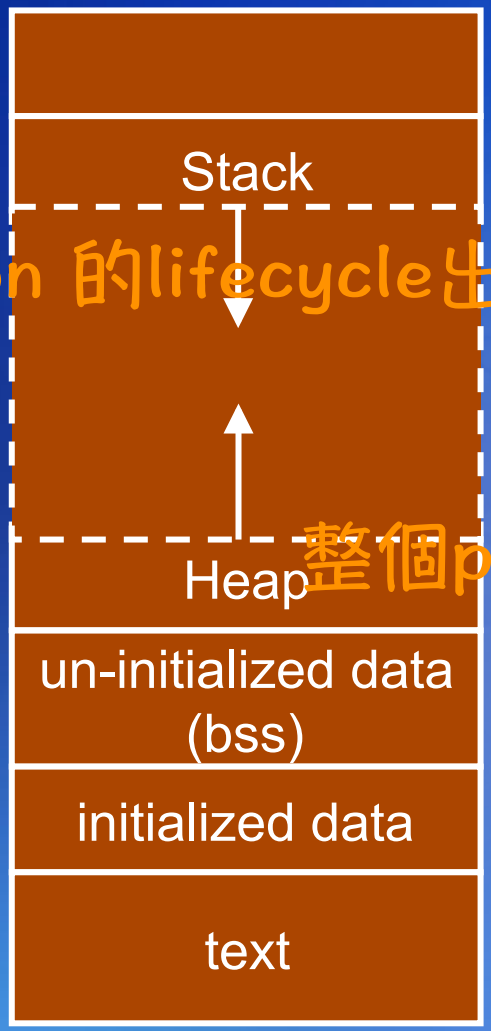
(Will be explained in details later; Ch8)

Services for Application Programs

- User Identification
- Process Management
- **Memory Management**
- File/Directory Management

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2008
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

Typical Memory Arrangement



command-line arguments and environment variables.

function 的lifecycle出了function就會被pop掉

整個program

Initialized to 0 by exec.

Read from program file by exec.

(Will be explained in details later; Ch7)

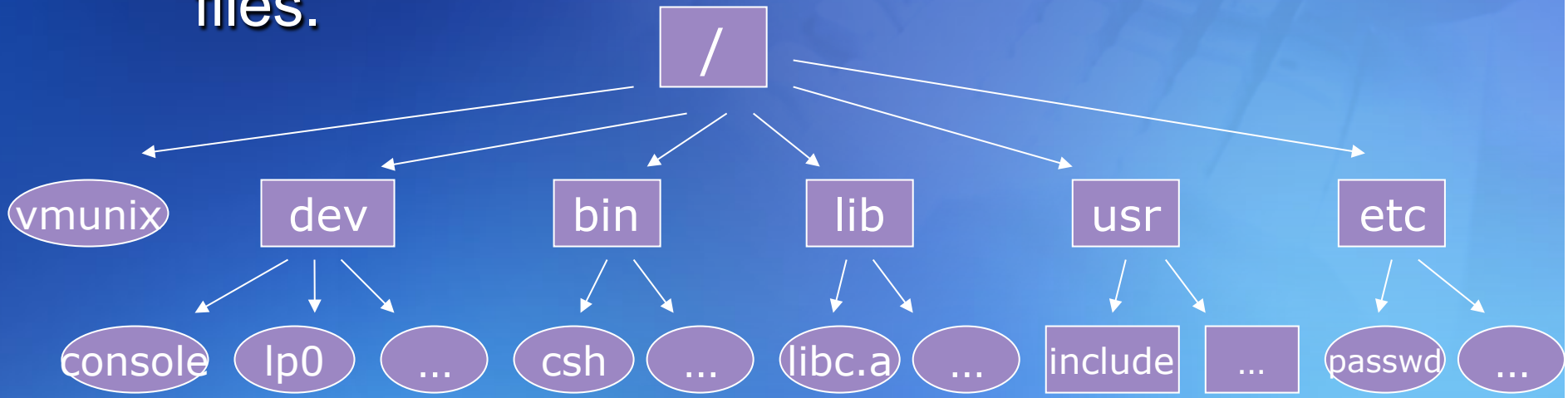
Tei-Wei Kuo, Chi-Sheng Shih and Hao-Hua Chu©2006
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

Services for Application Programs

- User Identification
- Process Management
- Memory Management
- File/Directory Management

UNIX File and Directory

- **File**
  - A sequence of bytes
- **Directory**
  - A file that includes info on how to find other files.

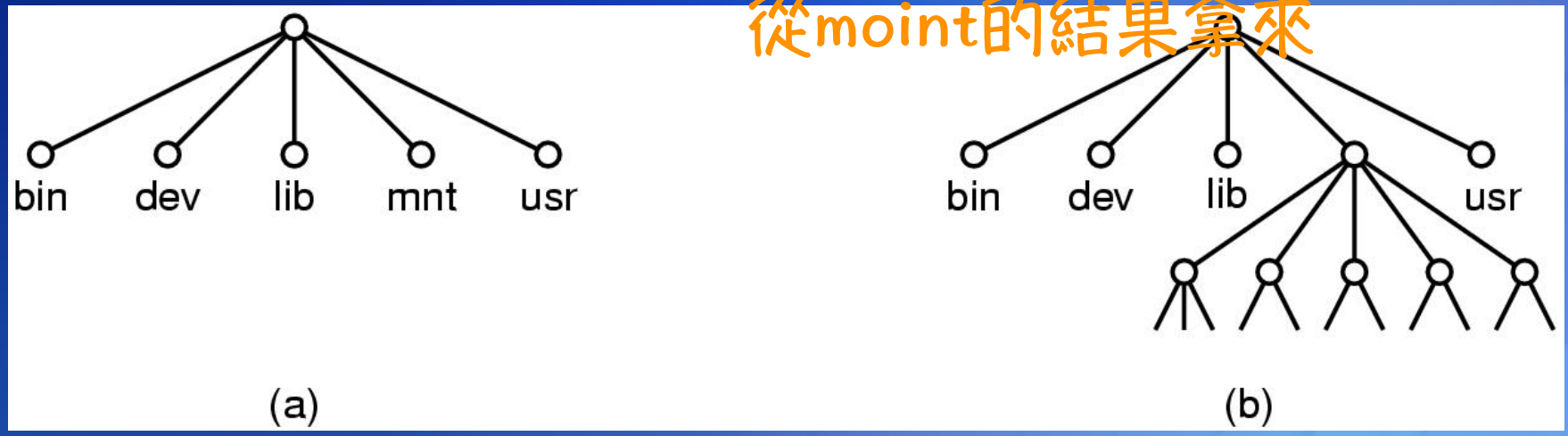


* Use command "mount" to show all mounted file systems!

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2007
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

Mount File Systems

自己的disk就有自己的file system
把新的fs，插進去（mount） /mnt/
從moint的結果拿來



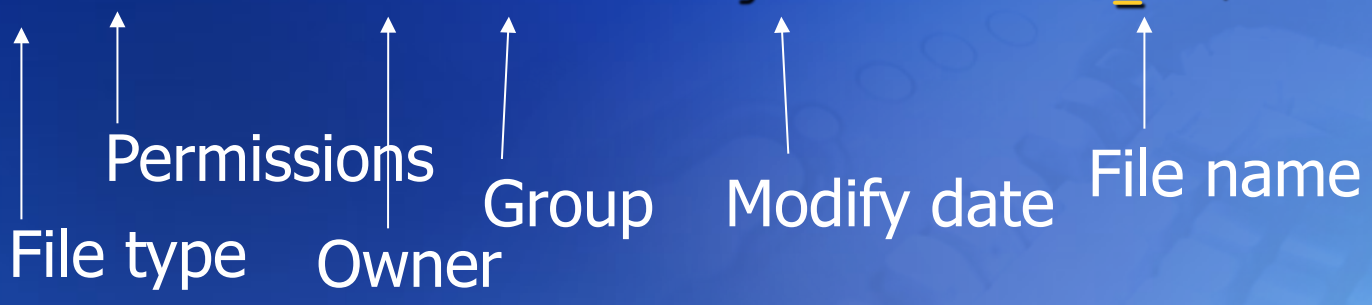
(a) File system before the mount

(b) File system after the mount

File Permission

Output of ls -l

```
total 4
lrwxr-xr-x 1 test user 18 Aug 28 13:41 home -> /usr/people/maria/
-rw-r--r-- 1 test user 94 Aug 28 13:42 nothing.txt
drwxr-xr-x 2 test user  9 Aug 28 13:40 test_dir/
```



Read (r) 4, write (w) 2, and execute (x) 1
For owner, group, and world (everyone)

```
chmod <mode> <file(s)>
chmod 700 file.txt
chmod g+rw file.txt
```

Related shell commands: chmod, chown, touch

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2007
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University

Some System Calls For Directory Management

Directory and file system management	
Call	**Description**
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

(Will be explained in details later; Ch4)

Some System Calls For File Management

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

(Will be explained in details later; Ch3)

What You Should Know

- **What is Operating System**
- **UNIX architecture**
- **Basic concepts of**
  - System call vs. function call
  - Kernel/user mode/space
  - Multitasking, time sharing
  - User identification
  - Process
  - Memory arrangement
  - Directory and file

把Chap 1的標題，
還有content背出來

Tei-Wei Kuo, Chi-Sheng Shih, Hao-Hua Chu, and Pu-Jen Cheng©2008
Department of Computer Science and Information Engineering
Graduate institute of Multimedia and Networking, National Taiwan University