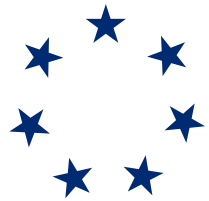
Introduction: Main OS Concepts

Instructor: Hengming Zou, Ph.D.

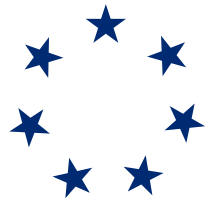
In Pursuit of Absolute Simplicity 求于至简，归于永恒

Content



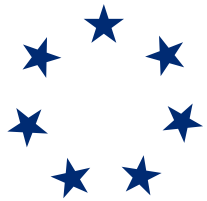
- Kernel and User Mode Programs
- Processes
- Memory
- Files
- System calls
- Shell

Kernel and User Mode Programs



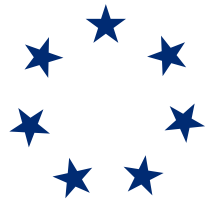
- Kernel Mode
  - Privilegeded mode
  - Strict assumptions about reliability/security of code
  - Memory resident
- User Mode:
  - More flexible
  - Simpler maintenance and debugging

Kernel and User Mode Programs



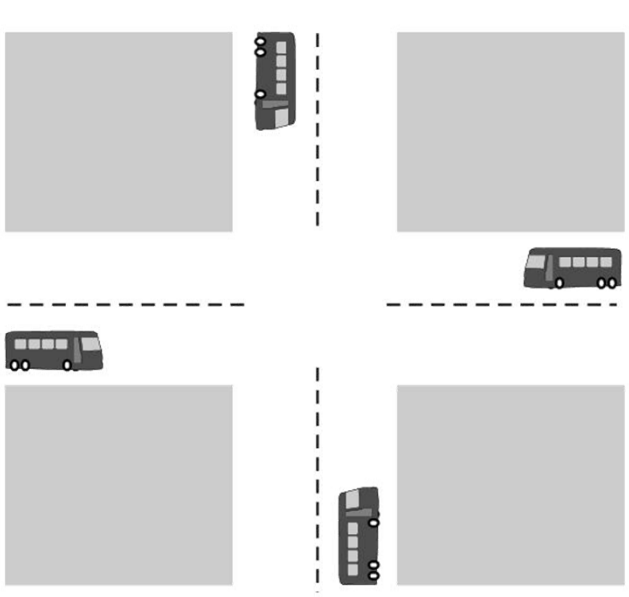
- Functionality Implemented in Kernel
  - CPU-, memory-, Input/Output managment
  - Multiprocessor management, diagnosis, test
  - Parts of file system and of the networking interface
- Functionality Implemented in User
  - Compiler, assembler, interpreter, linker/loader
  - File system management
  - Telecommunication
  - Network management
  - Editors, spreadsheets, user applications

Processes

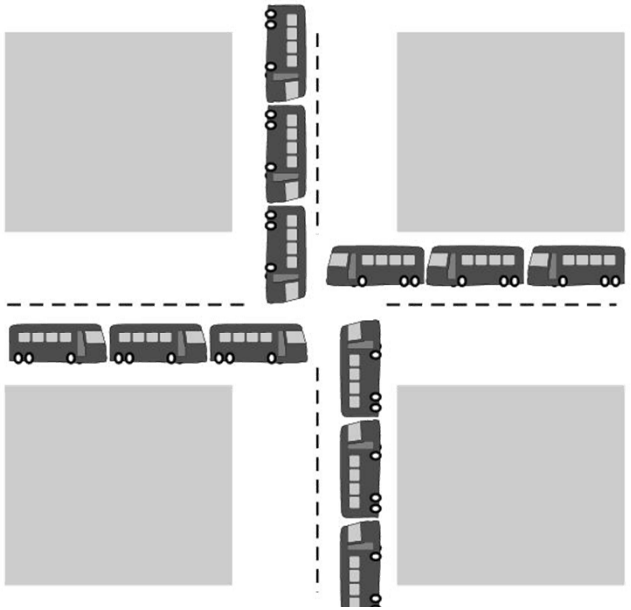


- Processes, process table, core image
- Parent and child processes
- Mutual exclusion and synchronization
- Inter-process communication
- Scheduling, signals
- Deadlock and synchronization

Processes

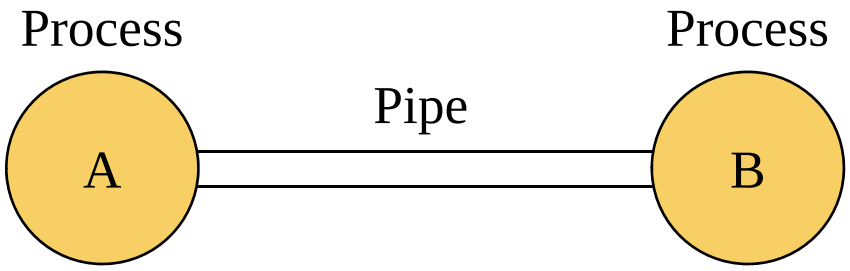


deadlock



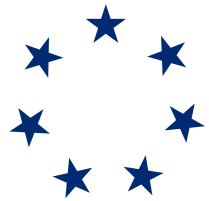
(a) A potential deadlock

(b) an actual deadlock



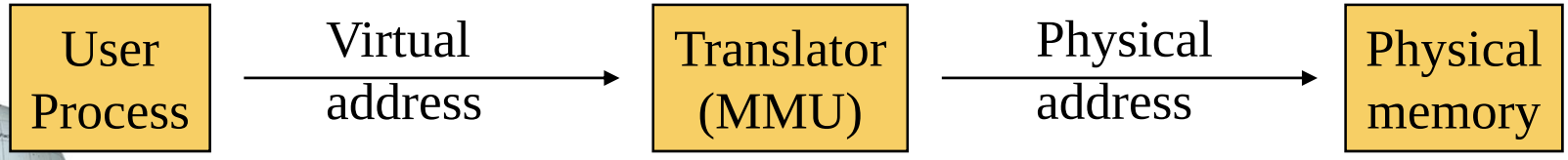
IPC

Memory Management

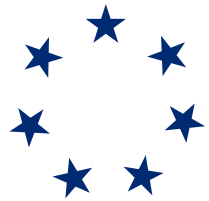


- Basic memory management
  - Segmentation, swapping, base and bound
- Virtual memory
  - Memory hierarchy, address translation
- Paging systems
  - Page replacement, design issues for paging systems
- Segmentation systems
  - Segmentation with paging

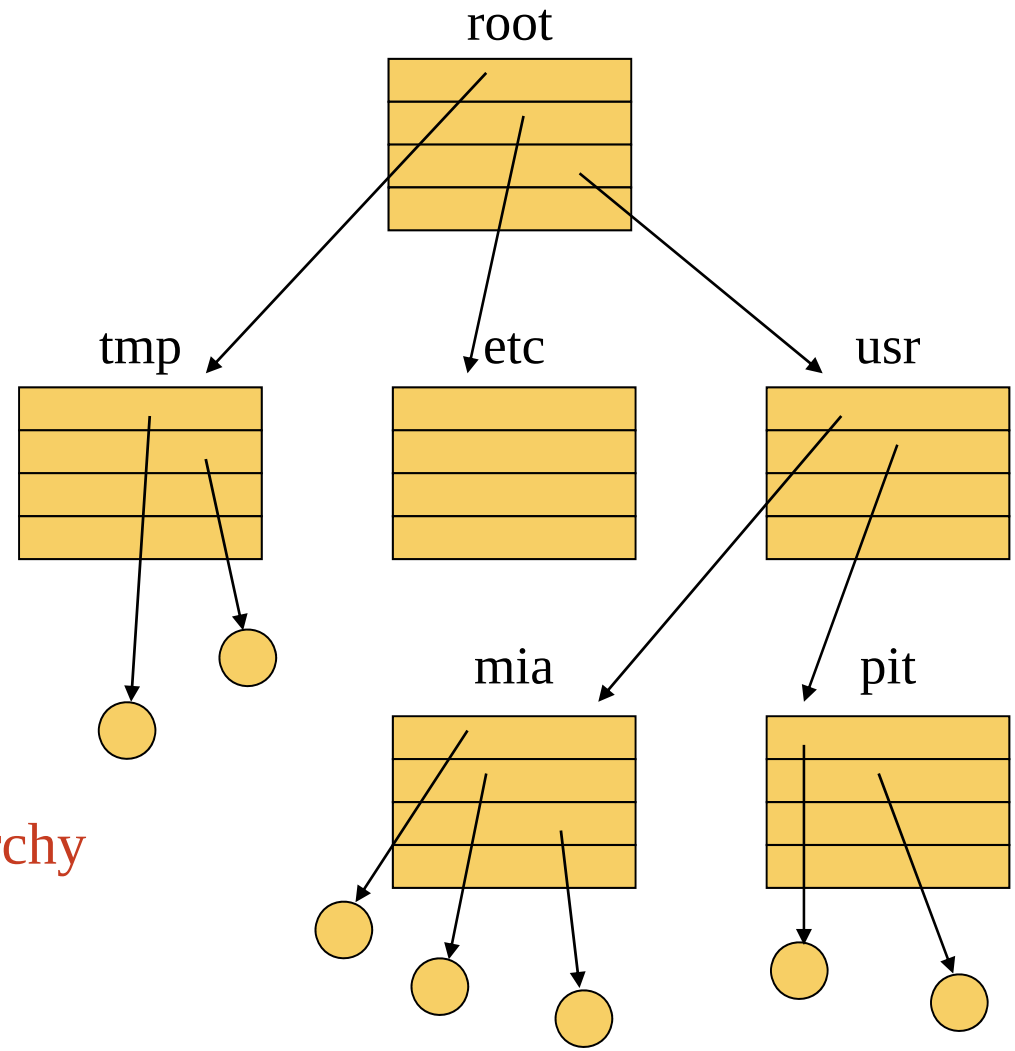
Dynamic Address Translation



Files

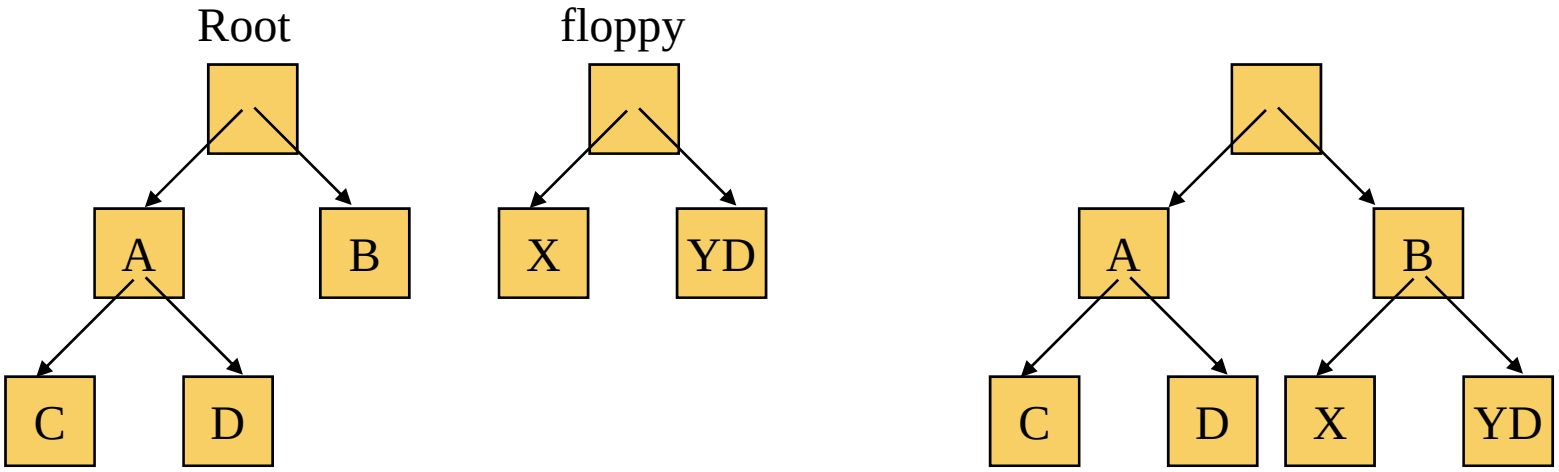


- Files, directories, root
- Path, working directory
- Protection, rwx bits
- File descriptor, handle
- Special files, I/O devices



A sample file hierarchy

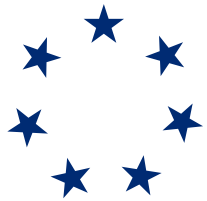
File Mounting



(a) (b)

- Before mounting,
  - files on floppy are inaccessible
- After mounting floppy on b,
  - files on floppy are part of file hierarchy

System Calls

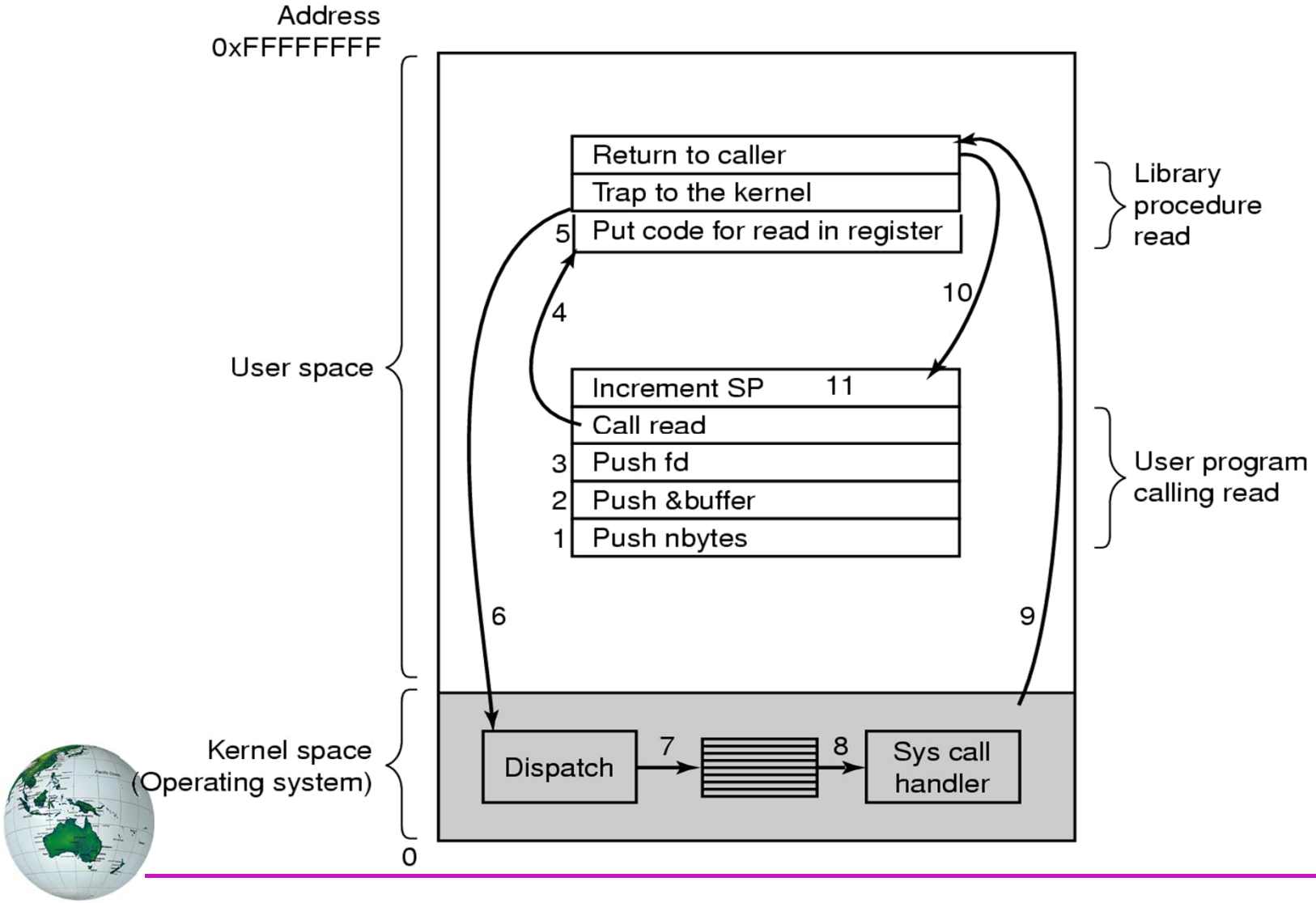


- User programs access operating system services via system calls
- Parameter transmission via trap, register, stack

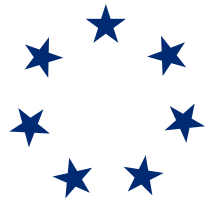
  count = read (file, buffer, nbytes);

- Five Classes of System Calls
  - Process control
  - File manipulation
  - Device manipulation
  - Memory manipulation
  - Information maintenance
  - Communications

Steps in Making a System Call

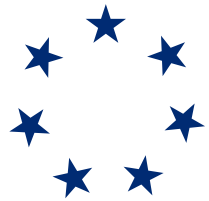


Sys Calls For Process Mgmt



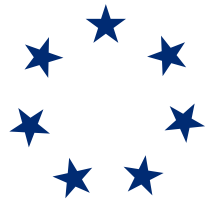
Call	Description
pid = fork()	Create a child process
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp	Replace a process's core image
exit(status)	Terminate process & return status

Sys Calls For File Mgmt



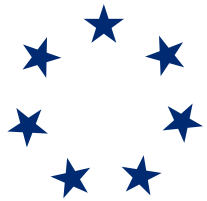
Call	Description
fd = open(file, how, …)	Open a file for reading or writing
s=close(fd)	Close an open file
n=read(fd, buffer, nbytes)	Read data from a file into a buffer
n=write(fd, buffer, nbytes)	Write data from buffer into a file
position=lseek(fd, offset, whence)	Move the file pointer
s = stat(name,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

Sys Calls For Directory Mgmt



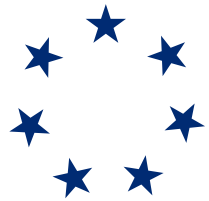
Call	Description
S = mkdir(name, mode)	Create a new directory
S = rmdir(name)	Remove an empty directory
S = link(name1, name2)	Create a entry name2 pointing to name1
S = mount(special, name, flag)	Mount a file system
S = umount(special)	Unmount a file system

Sys Calls For Miscellaneous Tasks



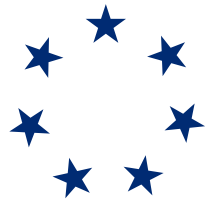
Call	Description
S = chdir(dirname)	Change the working directory
S = chmod(name, mode)	Change a file's protection bits
S = kill(pid, signal)	Send a signal to a process
Seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

Some Win32 API calls



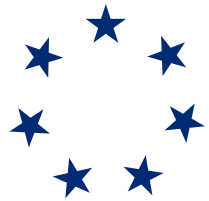
UNIX	Win32	Description
Fork	CreateProcess	Create a new process
Waitpid	WaitForSingleObject	Can wait for a process to exit
Execve	(none)	CreateProcess=fork+execve
Exit	ExitProcess	Terminate execution
Open	CreateFile	Create a file or open an existing file
Close	CloseHandle	Close a file
Read	ReadFile	Read data from a file
Write	WriteFile	Write data to a file
Lseek	SetFilePointer	Move the file pointer
Stat	GetFileAttributesEx	Get various file attributes

Some Win32 API calls



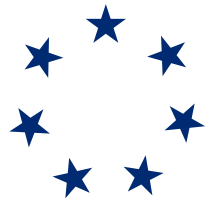
UNIX	Win32	Description
Mkdir	CreateDirectory	Create a new directory
Rmdir	RemoveDirectory	Remove an empty directory
Link	(none)	Win32 does not support links
Unlink	Deletefile	Destroy an existing file
Mount	(none)	Win32 does not support mount
Umount	(none)	Win32 does not support umount
Chdir	SetCurrentDirectory	Change the current working directory
Chmod	(none)	Win32 does not support security (NT does)
Kill	(none)	Win32 does not support signals
Time	GetLocalTime	Get the current time

Shell



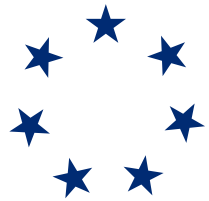
- Command interpreter
- Displays prompt
- Implements input/output redirection
- Background processes, job control
- Pseudo terminals

Shell Example



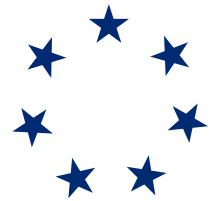
```
$ date

$ date >file

$ sort <file1 >file2

$ cat file1 file2 file3 > /dev/lp1

$ make all >log 2>&1 &
```

Implementing a Shell

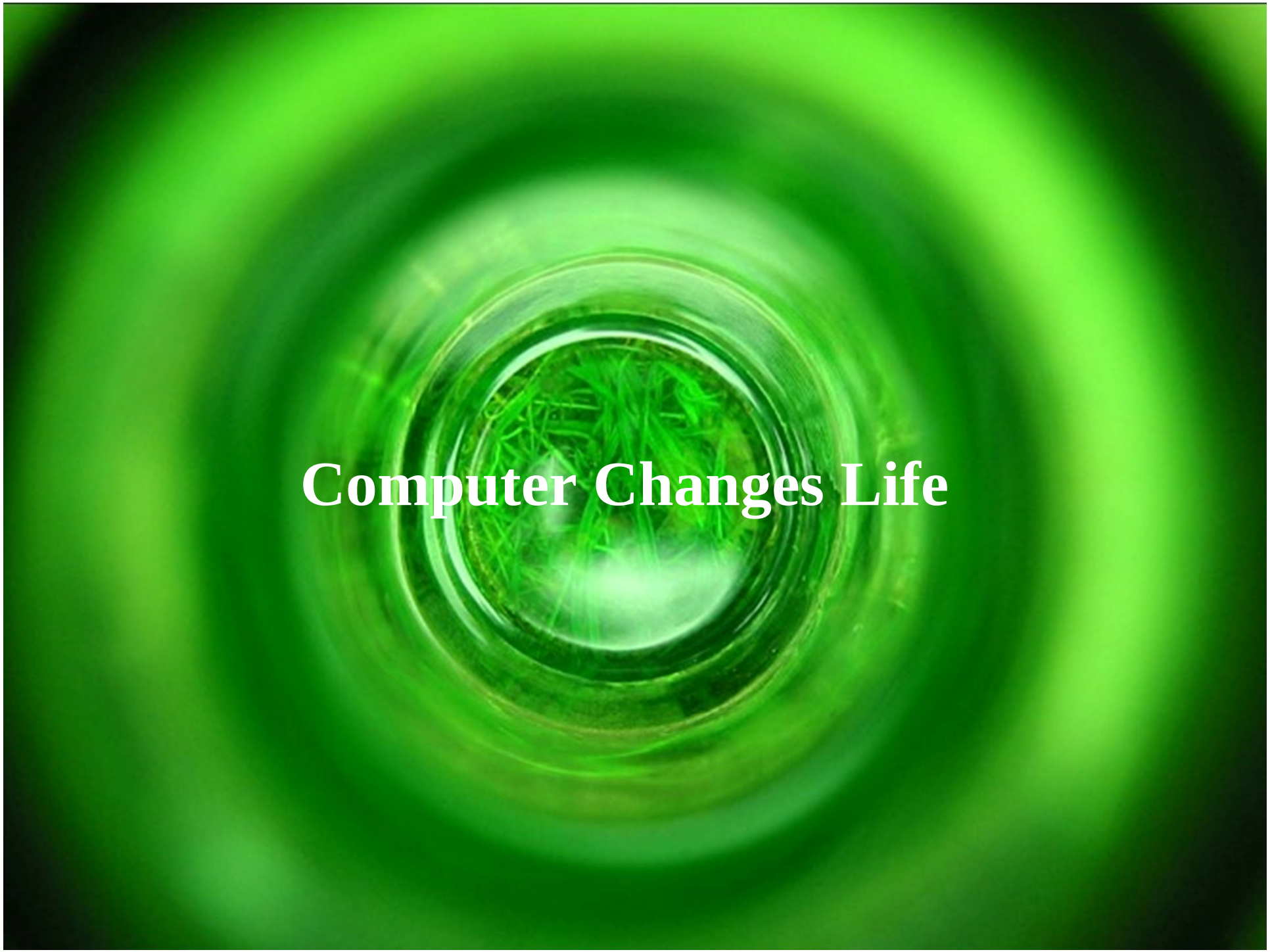


- Shell provides the user interface
  - sh, csh, tcsh, bash, zsh, etc.
- Windows Explorer is similar
  - looks like part of the operating system
  - but we now know enough to write a shell as a standard user program
- How to write a shell?

A Stripped Down Shell



```
while (TRUE) {                                  /* repeat forever */
    type_prompt( );                             /* display prompt */
    read_command (command, parameters)          /* input from terminal */
    if (fork() != 0) {                          /* fork off child process */
        /* Parent code */
        waitpid( -1, &status, 0);               /* wait for child to exit */
    }
    else {  /* Child code */
        execve (command, parameters, 0);        /* execute command */
    }
}
```


Computer Changes Life