Western

UNIVERSITY · CANADA

Chapter 2 - Operating System Structures

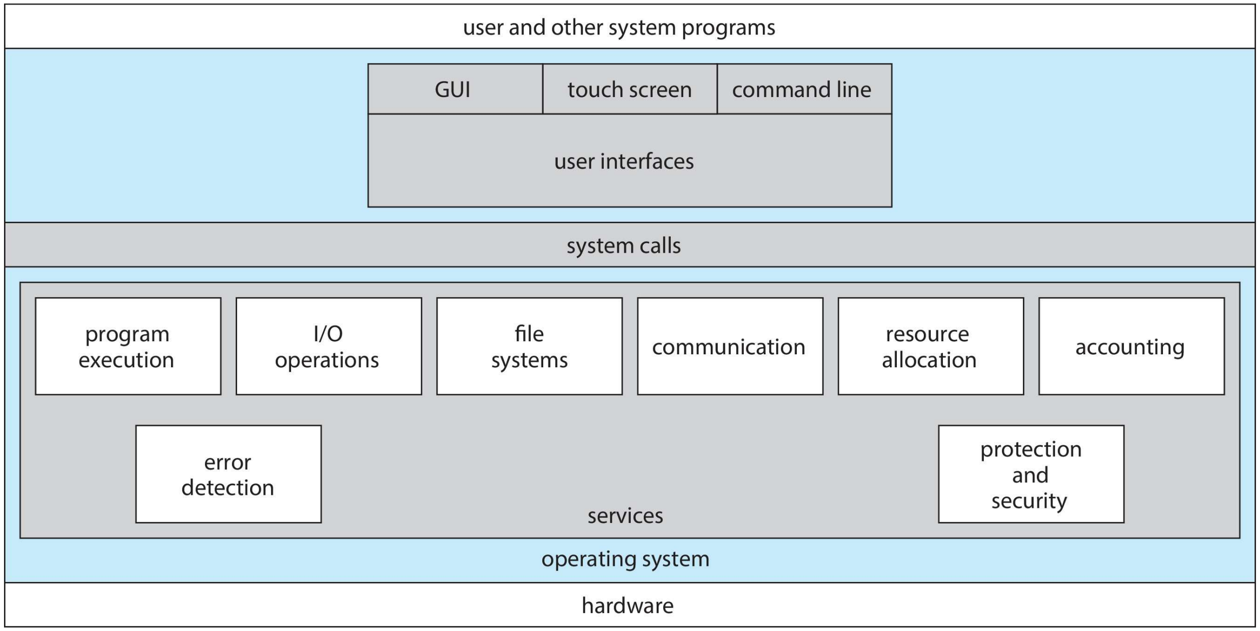
Spring 2023

Operating-System Structures

- Operating-System Services
- User and Operating-System Interface
- System Calls
- System Services
- Operating-System Structure
- Operating-System Debugging

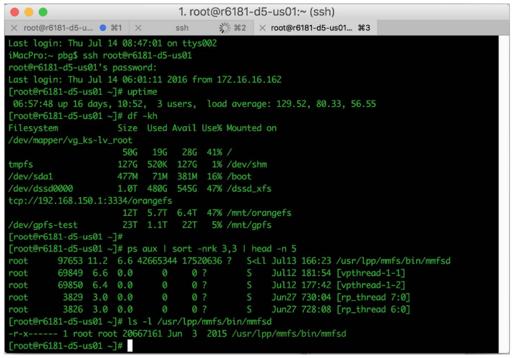
Operating-System Services

- Operating systems provide an environment for execution of programs and services to programs and users



User and Operating-System Interface

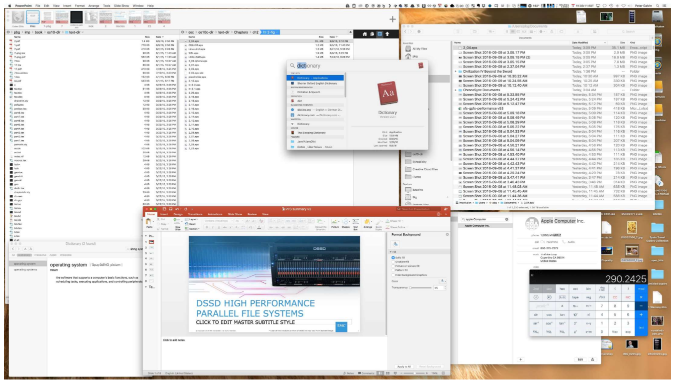
- Command-line Interface (**CLI**) allows direct command entry
- Sometimes implemented in kernel, sometimes by systems program
- Sometimes multiple flavors implemented – **shells**
- Primarily fetches a command from user and executes it
- Sometimes commands built-in, sometimes just names of programs
    - If the latter, adding new features doesn't require shell modification



```
Last login: Thu Jul 14 08:47:01 on ttys002
iMacPro:~ pbg$ ssh root@r6181-d5-us01
root@r6181-d5-us01's password:
Last login: Thu Jul 14 06:01:11 2016 from 172.16.16.162
[root@r6181-d5-us01 ~]# uptime
 06:57:48 up 16 days, 10:52,  3 users,  load average: 129.52, 80.33, 56.55
[root@r6181-d5-us01 ~]# df -kh
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg_ks-lv_root
                       50G   19G   28G  41% /
tmpfs                 127G  520K  127G   1% /dev/shm
/dev/sda1             477M   71M  381M  16% /boot
/dev/dssd0000         1.0T  480G  545G  47% /dssd_xfs
tcp://192.168.150.1:3334/orangefs
                       12T  5.7T  6.4T  47% /mnt/orangefs
/dev/gpfs-test         23T  1.1T   22T   5% /mnt/gpfs
[root@r6181-d5-us01 ~]#
[root@r6181-d5-us01 ~]# ps aux | sort -nrk 3,3 | head -n 5
root      97653 11.2  6.6 42665344 17520636 ?   S<Ll Jul13 166:23 /usr/lpp/mmfs/bin/mmfsd
root      69849  6.6  0.0      0     0 ?        S    Jul12 181:54 [vpthread-1-1]
root      69850  6.4  0.0      0     0 ?        S    Jul12 177:42 [vpthread-1-2]
root       3829  3.0  0.0      0     0 ?        S    Jun27 730:04 [rp_thread 7:0]
root       3826  3.0  0.0      0     0 ?        S    Jun27 728:08 [rp_thread 6:0]
[root@r6181-d5-us01 ~]# ls -l /usr/lpp/mmfs/bin/mmfsd
-r-x------ 1 root root 20667161 Jun  3  2015 /usr/lpp/mmfs/bin/mmfsd
[root@r6181-d5-us01 ~]# 
```

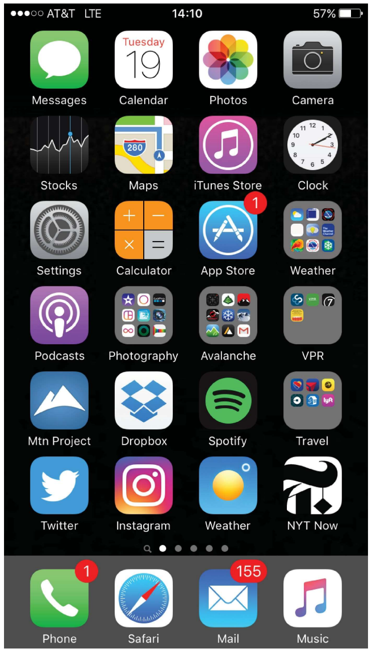
User and Operating-System Interface

- Graphical User Interface (**GUI**) is a user-friendly desktop metaphor interface
- Usually mouse, keyboard, and monitor
- Icons represent files, programs, actions, etc
- Various mouse buttons over objects in the interface
- Many systems now include both CLI and GUI interfaces


Dictionary
PFS summary v3
DSSD HIGH PERFORMANCE PARALLEL FILE SYSTEMS
CLICK TO EDIT MASTER SUBTITLE STYLE
Format Background
Apple Computer Inc.
290.2425

User and Operating-System Interface

- Touchscreen devices require new interfaces
  - Mouse not possible or not desired
  - Actions and selection based on gestures
  - Virtual keyboard for text entry
- Voice commands

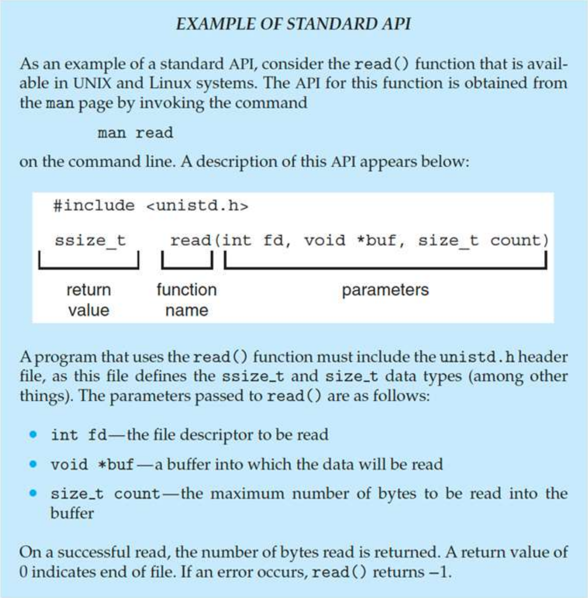


System Calls

- Programmers need their own interface for their programs
  - **System calls** are often used indirectly through a high-level Application Programming Interface (**API**)
    - Win32 API for Windows
    - POSIX API for POSIX-based systems
    - Java API for JVM
  - Typically written in C or C++. Sometimes in assembly when there is direct hardware access

System Calls

- The caller need know nothing about how the system call is implemented. Just needs to obey API and understand what OS will do as a result call. Most details of OS interface hidden from programmer by API



EXAMPLE OF STANDARD API

As an example of a standard API, consider the `read()` function that is available in UNIX and Linux systems. The API for this function is obtained from the `man` page by invoking the command

```
man read
```

on the command line. A description of this API appears below:

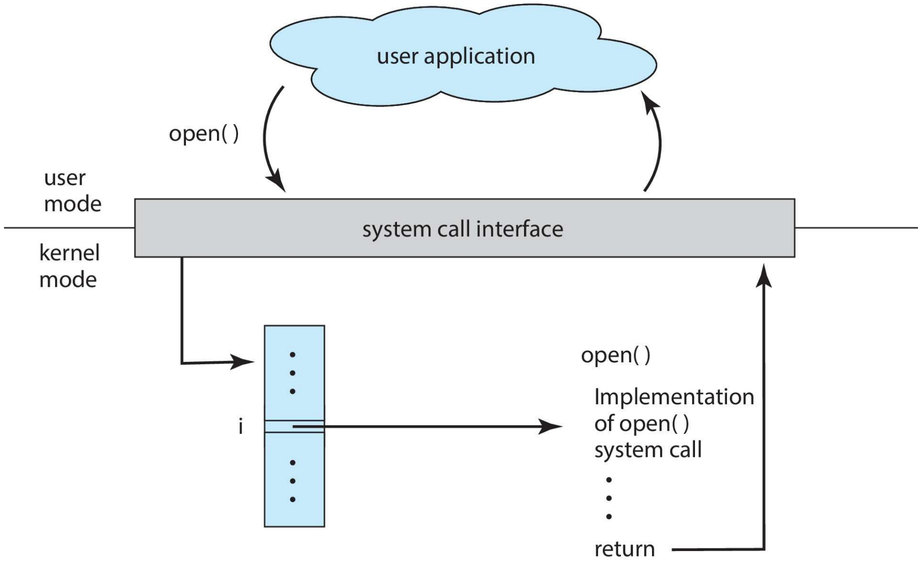
```
#include <unistd.h>

ssize_t     read(int fd, void *buf, size_t count)
```

return value · function name · parameters

A program that uses the `read()` function must include the `unistd.h` header file, as this file defines the `ssize_t` and `size_t` data types (among other things). The parameters passed to `read()` are as follows:

- `int fd`—the file descriptor to be read
- `void *buf`—a buffer into which the data will be read
- `size_t count`—the maximum number of bytes to be read into the buffer

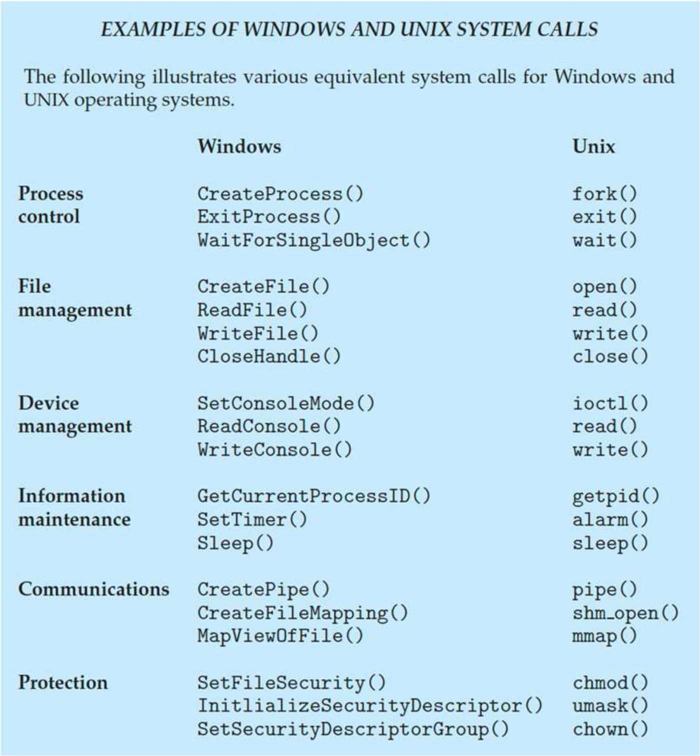
On a successful read, the number of bytes read is returned. A return value of 0 indicates end of file. If an error occurs, `read()` returns −1.



System Calls

- System call parameter passing
  - Often, more information is required than simply identity of desired system call
  - Three general methods used to pass parameters to the OS
    - **Registers**: Pass the parameters in registers. Simplest but there may be more parameters than registers (Linux uses this method if there are less than 6 parameters)
    - **Block**: Parameters stored in a block, or table, in memory, and address of block passed as a parameter in a register
    - **Stack**: Parameters placed, or pushed, onto the stack (in memory) by the program and popped off the stack by the operating system

System Calls



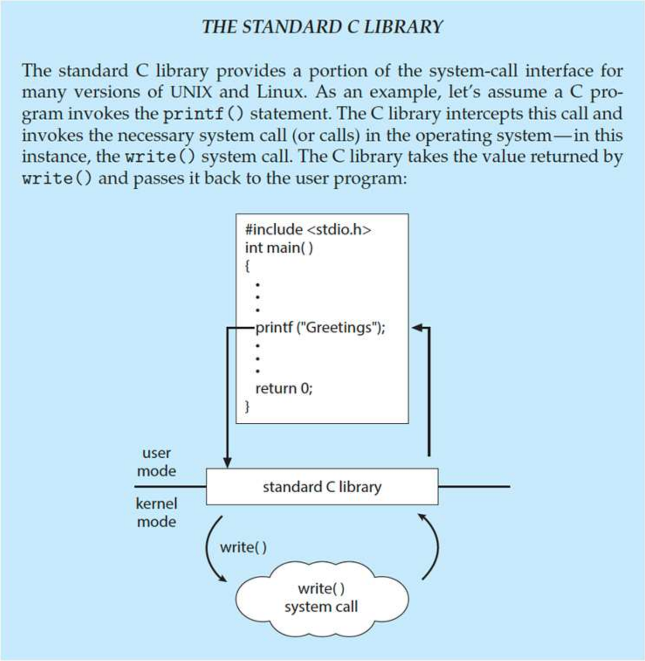
EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

The following illustrates various equivalent system calls for Windows and UNIX operating systems.

	Windows	Unix
Process control	`CreateProcess()`	`fork()`
	`ExitProcess()`	`exit()`
	`WaitForSingleObject()`	`wait()`
File management	`CreateFile()`	`open()`
	`ReadFile()`	`read()`
	`WriteFile()`	`write()`
	`CloseHandle()`	`close()`
Device management	`SetConsoleMode()`	`ioctl()`
	`ReadConsole()`	`read()`
	`WriteConsole()`	`write()`
Information maintenance	`GetCurrentProcessID()`	`getpid()`
	`SetTimer()`	`alarm()`
	`Sleep()`	`sleep()`
Communications	`CreatePipe()`	`pipe()`
	`CreateFileMapping()`	`shm_open()`
	`MapViewOfFile()`	`mmap()`
Protection	`SetFileSecurity()`	`chmod()`
	`InitlializeSecurityDescriptor()`	`umask()`
	`SetSecurityDescriptorGroup()`	`chown()`

System Calls

- Programmers usually use a system-call interface in an API which will use the system call



THE STANDARD C LIBRARY

The standard C library provides a portion of the system-call interface for many versions of UNIX and Linux. As an example, let's assume a C program invokes the `printf()` statement. The C library intercepts this call and invokes the necessary system call (or calls) in the operating system—in this instance, the `write()` system call. The C library takes the value returned by `write()` and passes it back to the user program:

```
#include <stdio.h>
int main( )
{
  .
  .
  .
  printf ("Greetings");
  .
  .
  .
  return 0;
}
```

System Services

- System programs provide a convenient environment for program development and execution.
- Most users’ view of the operating system is defined by system programs, not the actual system calls
- Provide a convenient environment for program development and execution
  - Some of them are simply user interfaces to system calls; others are considerably more complex

System Services

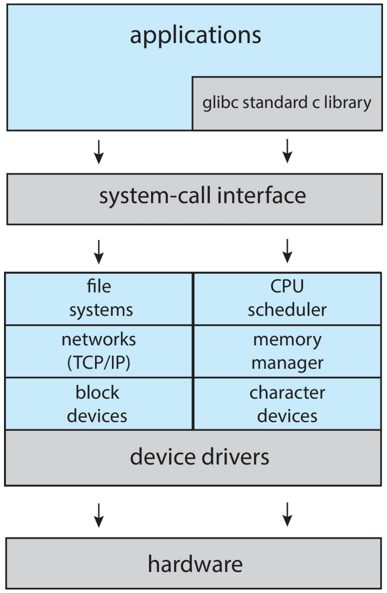
- E.g.
  - File manipulation
  - Status information sometimes stored in a file
  - Programming language support
  - Program loading and execution
  - Communications
  - Background services
  - Application programs

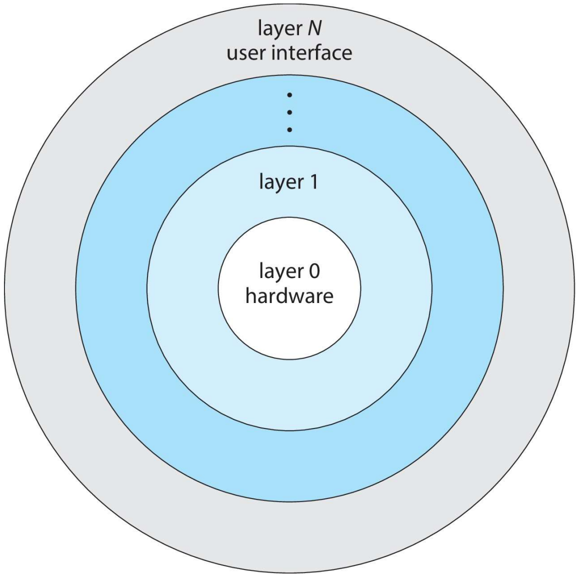
Operating-System Structure

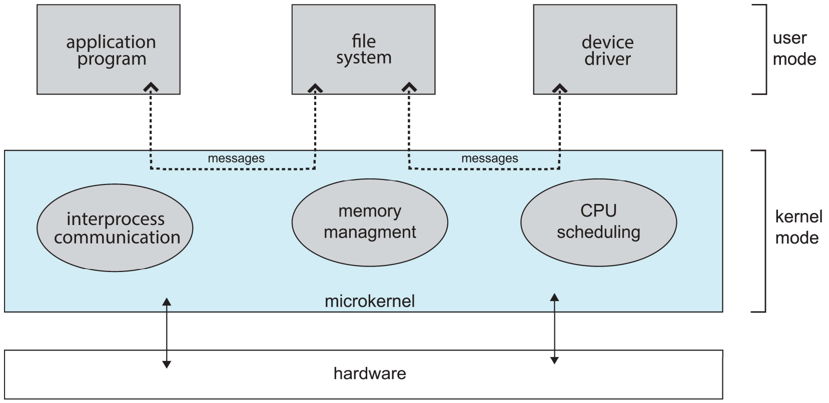
- General-purpose OS is very large program. Just as we do not put all our code in main(), operating systems usually divide functionality
  - Monolithic – Fast but hard to modify
  - Layered – Layers of modules from 0 (hardware) up to N (user interface)
  - Microkernels – Push as much functionality into user space as possible
  - Hybrid – Most operating systems today started as one of the above three and combined the other features. For efficiency and performance, most operating systems have shifted toward a "monolithic" structure with other structures added in

Operating-System Structure

- Monolithic vs. Layered vs. Microkernel







Operating-System Debugging

- OS generates **log files** containing error information
- Failure of an application can generate **core dump** file capturing memory of the process
- Operating system failure can generate **crash dump** file containing kernel memory

Operating-System Debugging

- OS must provide means of computing and displaying measures of system behavior
  - E.g. Counters
    - Windows
      - Windows Task Manager
    - Linux has several tools:
      - System wide: vmstat, netstat, iostat, perf
      - Per Process: ps, top

Operating-System Debugging

- OS collects data for a specific event, such as steps involved in a system call invocation
    - E.g. Tracing
        - Windows
            - Windows Sysinternals
        - Linux has several tools:
            - System wide: perf, tcpdump
            - Per Process: strace, gdb

Western

UNIVERSITY · CANADA