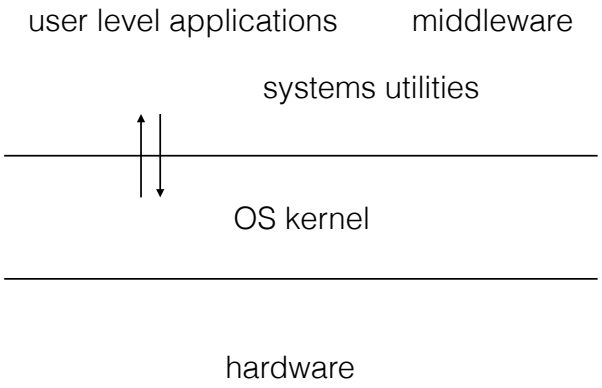
# Operating systems

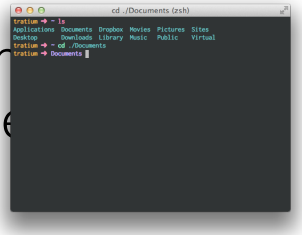
Structure and services

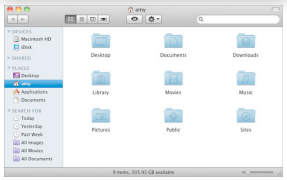


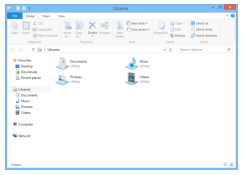

## Ex: writing from one file to another

- get input from user
- open files
- read from file
- write to file
- close files

## Ex: writing from one file to another



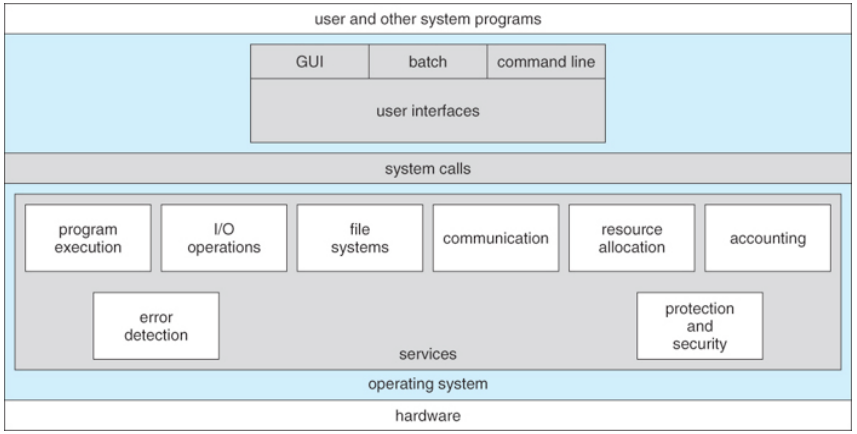




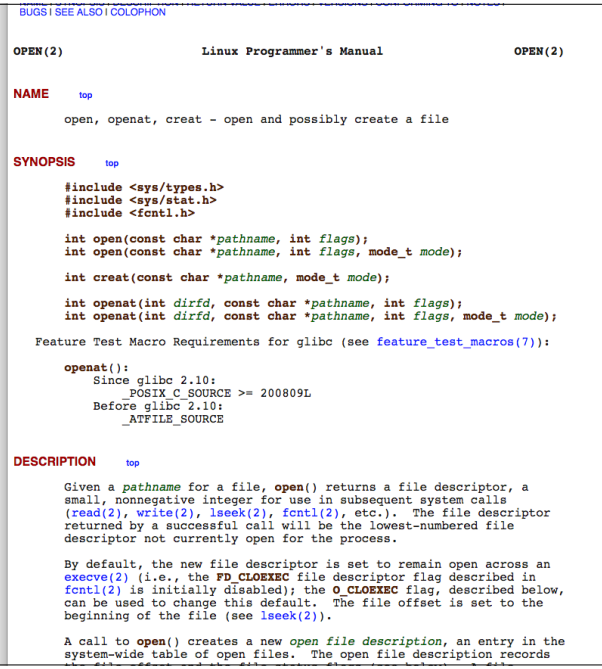
- get input from user
- open files
- read from file
- write to file
- close files

# Ex: writing from one file to another

- get input from user — I/O service
- open files — file system
- read from file
- write to file — errors
- close files




| | Windows | Unix |
|---|---|---|
| Process Control | CreateProcess()<br>ExitProcess()<br>WaitForSingleObject() | fork()<br>exit()<br>wait() |
| File Manipulation | CreateFile()<br>ReadFile()<br>WriteFile()<br>CloseHandle() | open()<br>read()<br>write()<br>close() |
| Device Manipulation | SetConsoleMode()<br>ReadConsole()<br>WriteConsole() | ioctl()<br>read()<br>write() |
| Information Maintenance | GetCurrentProcessID()<br>SetTimer()<br>Sleep() | getpid()<br>alarm()<br>sleep() |
| Communication | CreatePipe()<br>CreateFileMapping()<br>MapViewOfFile() | pipe()<br>shmget()<br>mmap() |
| Protection | SetFileSecurity()<br>InitlializeSecurityDescriptor()<br>SetSecurityDescriptorGroup() | chmod()<br>umask()<br>chown() |



BUGS | SEE ALSO | COLOPHON  Search online pages

OPEN(2)  Linux Programmer's Manual  OPEN(2)

NAME top

open, openat, creat - open and possibly create a file

SYNOPSIS top

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);

int creat(const char *pathname, mode_t mode);

int openat(int dirfd, const char *pathname, int flags);
int openat(int dirfd, const char *pathname, int flags, mode_t mode);
```

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

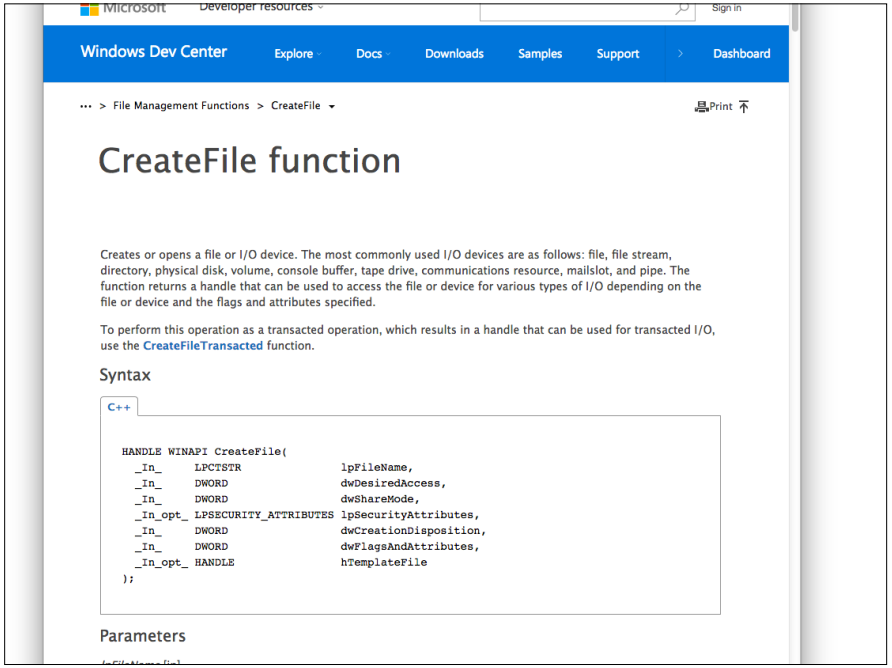
openat():
Since glibc 2.10:
_POSIX_C_SOURCE >= 200809L
Before glibc 2.10:
_ATFILE_SOURCE

DESCRIPTION top

Given a *pathname* for a file, **open**() returns a file descriptor, a small, nonnegative integer for use in subsequent system calls (read(2), write(2), lseek(2), fcntl(2), etc.). The file descriptor returned by a successful call will be the lowest-numbered file descriptor not currently open for the process.

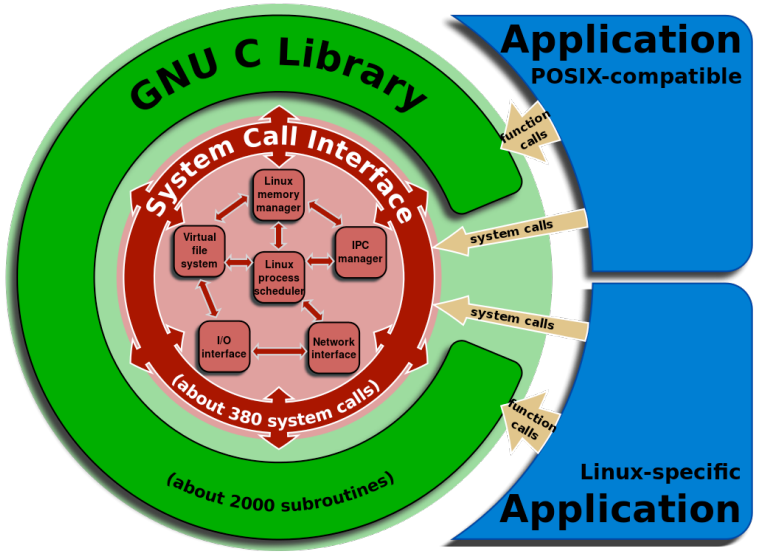
By default, the new file descriptor is set to remain open across an execve(2) (i.e., the **FD_CLOEXEC** file descriptor flag described in fcntl(2) is initially disabled); the **O_CLOEXEC** flag, described below, can be used to change this default. The file offset is set to the beginning of the file (see lseek(2)).

A call to **open**() creates a new *open file description*, an entry in the system-wide table of open files. The open file description records

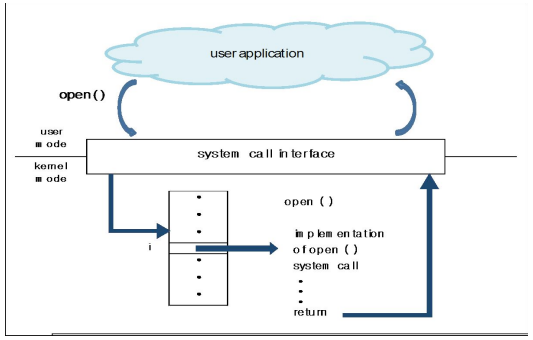



# Common APIs

- POSIX API
  - UNIX, Linux, Mac OS X
- Win32 API
- Java API

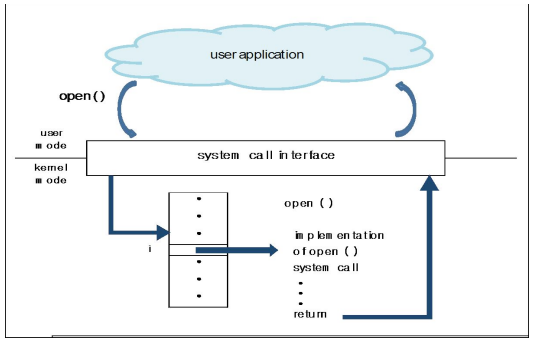



# Making a system call

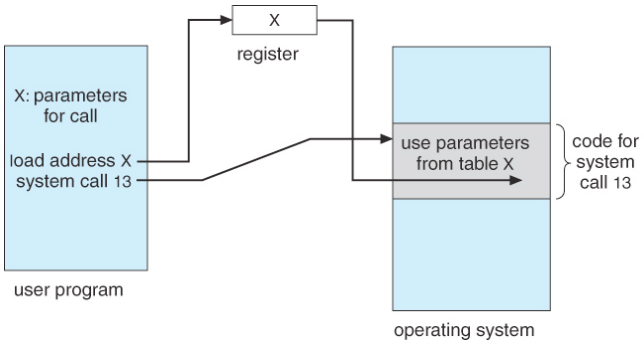



- typically in C or C++
- in assembly for low-level tasks

# Making a system call



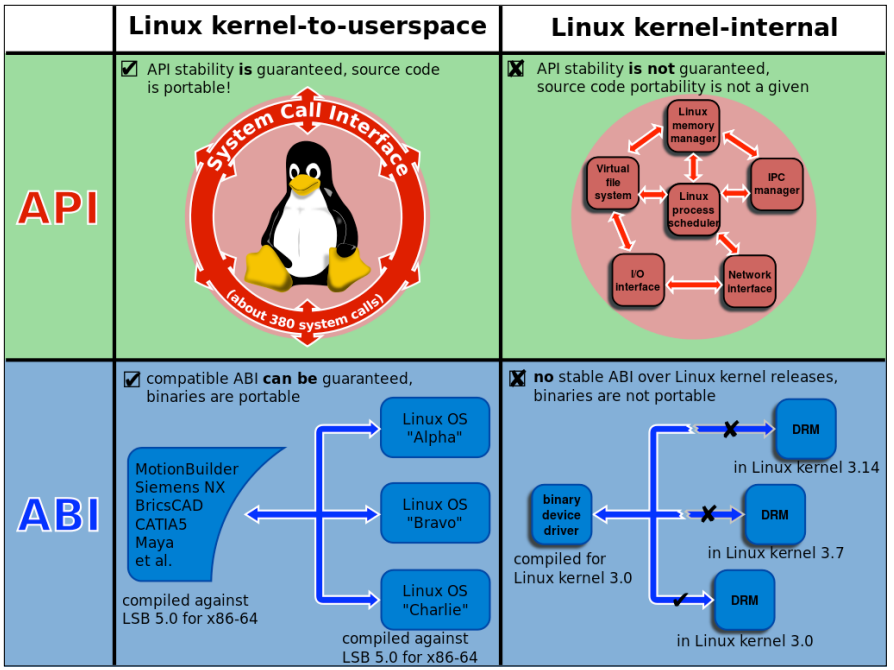

- 3 ways to pass parameters:
- registers
- block or table
- program stack




Passing of parameters as a table

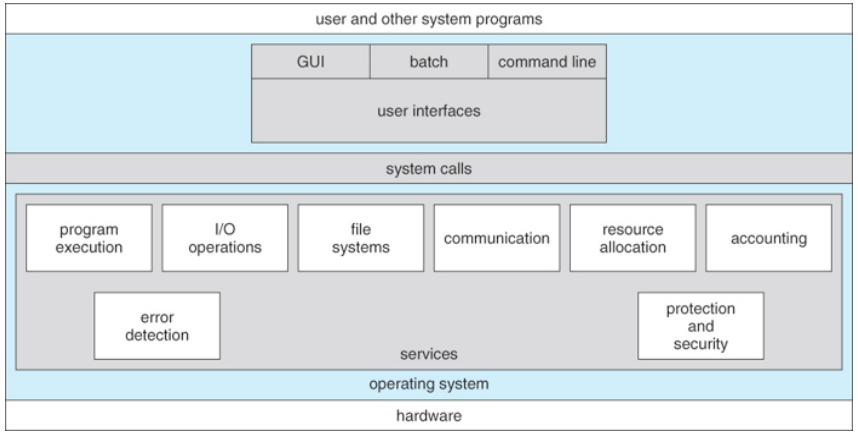
# Accessing OS services

- system calls
- libraries
- message passing
- system services



# Interoperability

- Can't test all applications on all platforms
- Focus on well-designed interfaces
- Test for compliance
- APIs need must be stable, backwards compatible




# User interface

- Batch processing
- Command-line
- GUI

# GUIs in Linux

- desktop environment (KDE, gnome, Unity, XFCE)
- WIMP: window, icon, menu, pointer
- modularity: ex choose your favorite lock-screen
- display server: responsible for input output
- window manager: tiling (i3, monad, open box)
- compositor (Compiz)