L5: System calls, types of system calls, system programs

Dr. Rajashree Dash

Associate Professor,
Department of CSE,
ITER, Siksha O Anusandhan Deemed to be University

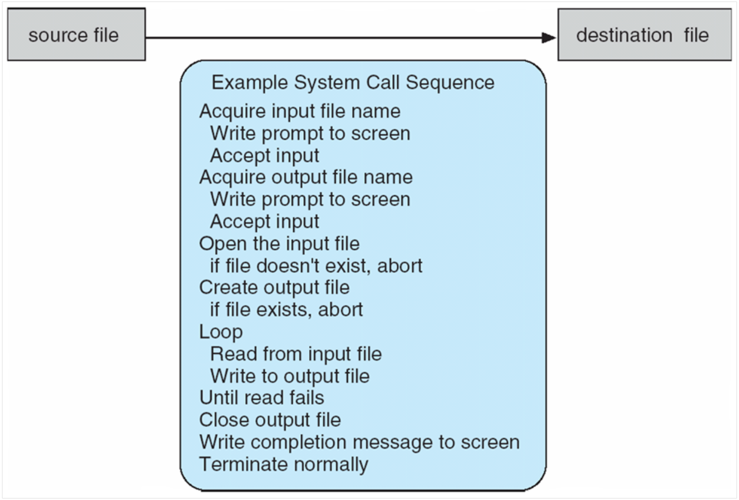
Outline

System calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level Application Programming Interface (API) rather than direct system call use

System calls

System call sequence to copy the contents of one file to another file



```
Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally
```

Figure: Example of use of system call

Application Programming Interface

- Typically, application developers design programs according to an application programming interface (API).
- The API specifies a set of functions that are available to an application programmer, including the parameters that are passed to each function and the return values the programmer can expect.
- Three of the most common APIs available to application programmers are:
  - Windows API for Windows systems
  - POSIX API for POSIX-based systems (which include virtually all versions of UNIX, Linux, andMac OSX)
  - Java API for programs that run on the Java virtual machine.
- A programmer accesses an API via a library of code provided by the operating system. In the case of UNIX and Linux for programs written in the C language, the library is called libc.

Application Programming Interface

- 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 >

  $ssize\_t\ read(int\ fd, void * buf, size\_t\ count)$

  The parameters passed to read() are as follows:

  • int fd — the file descriptor to be read

  • void *buf — a buffer where the data will be read into

  • 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.

Run time support system

- It is a set of functions built into libraries included with a compiler.
- For most programming languages , run time support system provides a system call interface that serves as the link to system calls made available by the operating system.
- The system call interface intercepts function calls in the API and invokes the necessary system calls within the operating system.

API - System call - OS relationship

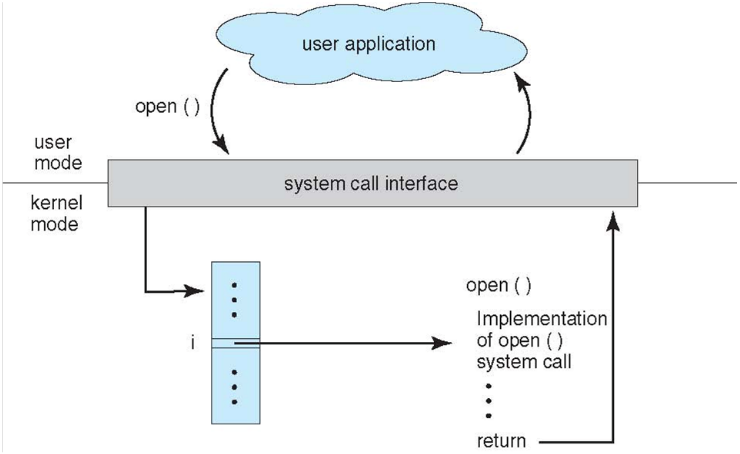
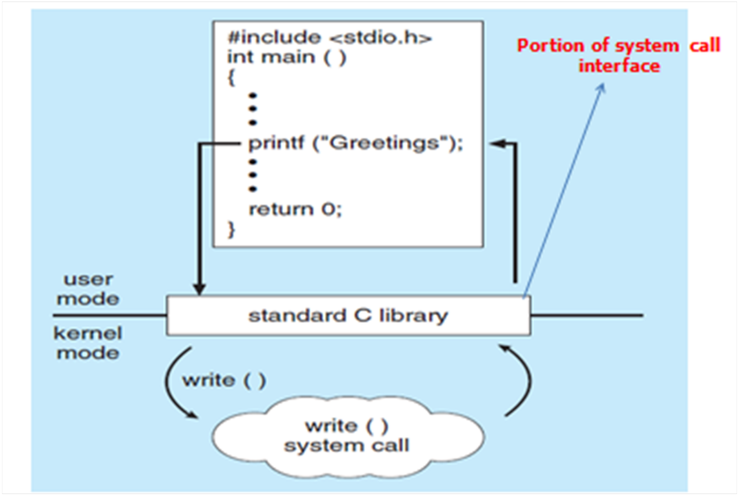


Figure: API - System call - OS relationship

API - System call - OS relationship



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

Figure: API - System call - OS relationship

System call implementation

- Typically, a number is associated with each system call.
- System-call interface maintains a table indexed according to these numbers.
- The system call interface invokes the intended system call in OS kernel and returns status of the system call and any return values
- The caller needs to know nothing about how the system call is implemented or what it does during execution.
- It 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

System call parameter passing

Three general methods are used to pass parameters to the OS:

- Simplest: pass the parameters in registers.
- Parameters stored in a block, or table, in memory, and address of block is passed as a parameter in a register. (Used in Linux and Solaris)
- Parameters is placed, or pushed, onto the stack by the program and popped off the stack by the operating system.

Block and stack methods do not limit the number or length of parameters being passed.

Types of System calls

- Process control
  - create process, terminate process
  - end, abort
  - load, execute
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory
  - Dump memory if error
  - Debugger for determining bugs, single step execution
  - Locks for managing access to shared data between processes
- File management
  - create file, delete file
  - open, close file
  - read, write, reposition
  - get and set file attributes

Types of System calls

- Device management
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices
- Information maintenance
  - get time or date, set time or date
  - get system data, set system data
  - get and set process, file, or device attributes

Types of System calls

- Communications
  - create, delete communication connection
  - send, receive messages if message passing model to host name or process name
  - Shared-memory model create and gain access to memory regions
  - transfer status information
  - attach and detach remote devices
- Protection
  - Control access to resources
  - Get and set permissions
  - Allow and deny user access

System Programs

System programs also known as system utilities provide a convenient environment for program development and execution. Different categories of system program:

- File management
  Create, delete, copy, rename, print, list, and generally manipulate files and directories.
- Status information
  Some programs simply ask the system for info - date, time, amount of available memory, disk space, number of users or similar status information. Others provide detailed performance, logging, and debugging information. Typically, these programs format and print the output to the terminal or other output devices. Some systems implement a registry that is used to store and retrieve configuration information.

System Programs

- File modification
  Text editors to create and modify files. Special commands to search contents of files or perform transformations of the text
- Programming-language support
  Compilers, assemblers, debuggers and interpreters for common programming languages are often provided with the OS or available as a separate download.
- Program loading and execution
  Absolute loaders, reloadable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language.

System Programs

- Communications
  Provide the mechanism for creating virtual connections among processes, users, and computer systems Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another
- Background Services
  • Launch at boot time
    - Some for system startup, then terminate
    - Some from system boot to shutdown

  • Provide facilities like disk checking, process scheduling, error logging, printing • Run in user context not kernel context