**SYSTEM CALLS/ API or SYSTEM PROGRAMS**

**SYSTEM CALLS**

System calls provide an interface to services made available by Operating System. These calls are generally available as routines written in C and C++.

How it works?

Writing a simple program to read data from one file and copy them to another file. On mouse based system and icon based system, a menu of file name is usually displayed in a window. The user can then use mouse to select source name and window can be opened for destination name to specified. This sequence requires I/O system calls.

Once two file names are obtained program must open input file and create output file. Each of these operations require another system calls.

Error: when program tries to open input file, it may find that there is no file of that name or that the file is protected against access. In these cases, program should print a message (sequence of system calla) and then terminate abnormally (another system call).

SOURCE FILE

DESTINATION

**Example System Call Sequence**

Acquire input file name

Write prompt to screen (system call)

Accept input

Acquire output file name

Write prompt to screen

Accept input

Open input file

If file doesn’t exist, abort

Create output file

If file exist, abort

Loop

Read from input file

Write to output file

Until read fails

Close output file

Write completion message on screen

Terminate normally

Press key to keyboard for output

Interaction

between

O.S. and

Process

**Code for**

**system call 13**

Use parameters from table X

x

register

X: Parameters for call

Load address X

System call 13

**There are three methods to pass parameters to O.S.**

* Pass the parameters in registers.
* In these cases, parameters are generally stored in a block or table in memory, and the addresses of the block is passed as parameter in register.
* Some O.S. prefer stack methods because they do not limit the number or length of parameters passed.

**Types of System Calls**

1. **Process Control**

A running program needs to be able to halt its execution either normally (End) or abnormally (aboard). If a system call is made to terminate the currently running program abnormally, or if the program runs into a problem and causes an error trap, a dump of memory is sometimes taken and an error message is generated.

It performs the following actions:

* End, abort
* Load, execute
* Create, terminate process
* Get process attributes, set process attributes
* Wait for time
* Wait event, signal event
* Allocate free memory

1. **File Management**

It performs the following actions:

* Create file, delete file
* Open, close
* Read, write, read position
* Get file attributes, set file attributes

1. **Device Management**

* Request device, release device
* Read, write, reposition
* Get device attributes, set device attributes
* Logically attach or detach devices

1. **Information Maintenance**

Operating system keeps the information about all its processes and there are system calls to access this information.

This performs the following actions:

* Get time or date, set time or date
* Get system data, set system data
* Get process, file, or device attributes
* Set process, file, or device attributes

1. **Communications**

There are two models of communication.

* Message Passing Model

In this information is exchanged through an inter-process communication facility provided by the O.S.

* Shared Memory Model

In this processes use map memory system calls to gain access to regions of memory owned by other processes.

This (communication) system performs following actions:

* Create, delete communication connection.
* Send, receive messages.
* Transfer status information.
* Attach or detach remote devices.

**SYSTEM PROGRAMS**

They provide a convenient environment for program development and execution.

**They can be divided into six categories.**

1. **File Management**

These programs create, delete, copy, rename, dump, and generally manipulate files and directories.

1. **Status Information**

Some programs ask for system data, time, amount of available memory or disk space, number of users. These programs format and print the output to terminal or display it in the window of GUI (graphical user interface).

1. **File Modification**

There are several text editors used to create and modify the contents of files stored on disk or other storage devices. There are special commands to search contents of files or perform transformation of the text.

1. **Programming Language Support**

Compilers, assemblers, debuggers and interpreters for common programming languages are provided with operating system.

1. **Program Loading and Execution**

Once a program is assembled or combined, it must be loaded into memory to be executed.

1. **Communications**

Provide mechanism for creating virtual connections among processes, users and computer systems. They allow users to send messages to one another’s screens, to browse web pages, to send emails to log in remotely or to transfer files.

**APPLICATION PROGRAMMING INTERFACE (API)**

It specifies the set of functions that are available to an application programmer, including the parameters that are passed to each function and the return values programmer can expect.

Three most common API’s are:

* Win 32 for Windows.
* POSIX API for POSIX based systems (for all versions of UNIX, LINUX, MAC OSX).
* Java API for designing programs that run on JVM (java virtual machine).