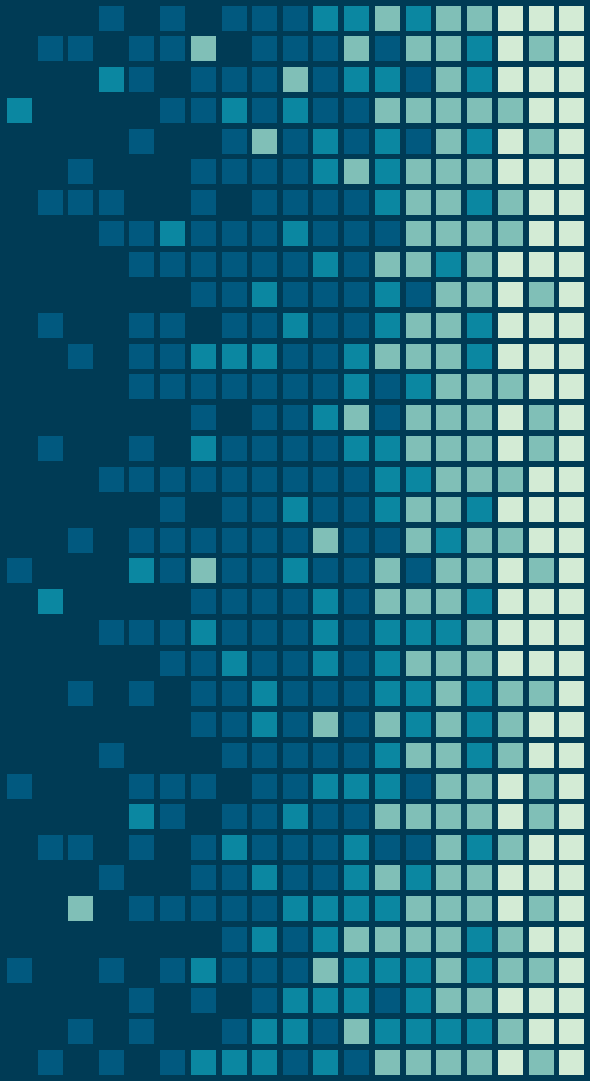
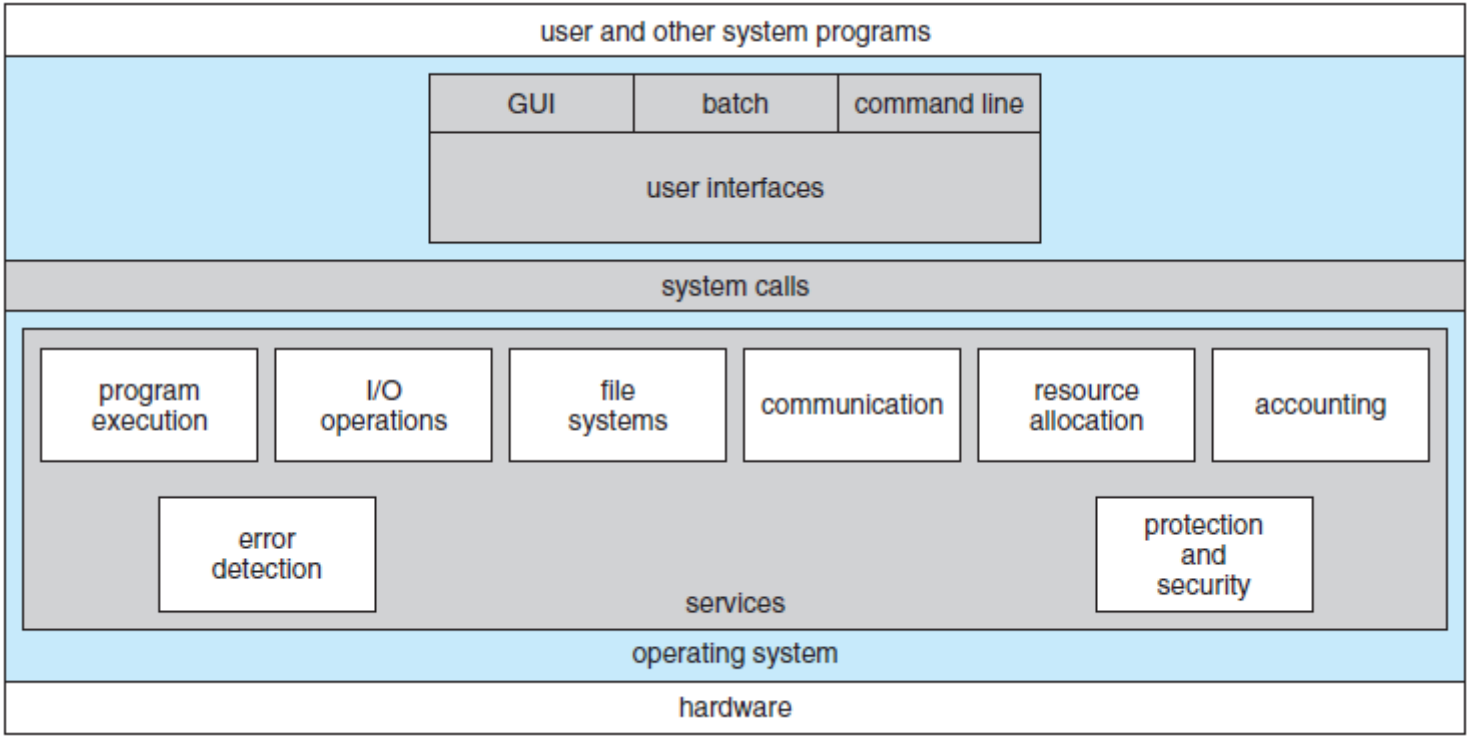
Sistemas Operativos I

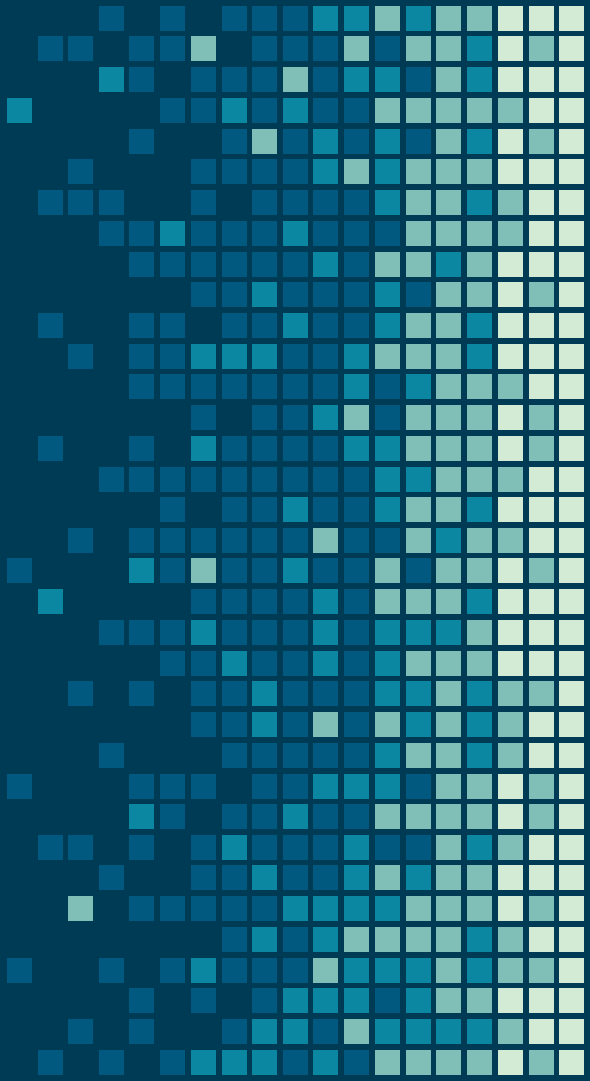
System Structures



System Services



Services



- **User interface**. (UI). Command-line interface (CLI), batch interface, y graphical user interface (GUI).
- **Program execution**.. Cargar el programa a memoria y de alguna manera ser capaz de detenerlo ya sea de una manera controlada o forzada.
- **I/O operations**.. Dispositivos internos y externos. teclados, DVDs, etc.
- **File-system manipulation**.. Manipulación de archivos y directorios.
- **Communications**.. Comunicación entre unos procesos y otros. Puede hacerse por memoria compartida o mensajes compartidos.
- **Error detection**... Al ser un sistema completo, debe de guardar y corregir errores que ocurran. Errores de pariedad, comunicación, etc.

- **Resource allocation**.. Asignación de recursos cuando múltiples usuarios están usando el mismo sistema.
- **Accounting**.. Guardar la cantidad de recursos utilizados del Sistema. Ejemplo cobro de servicios cloud.
  Protection and security.. Procesos que no puedan alterar archivos de otros usuarios o sin permisos.

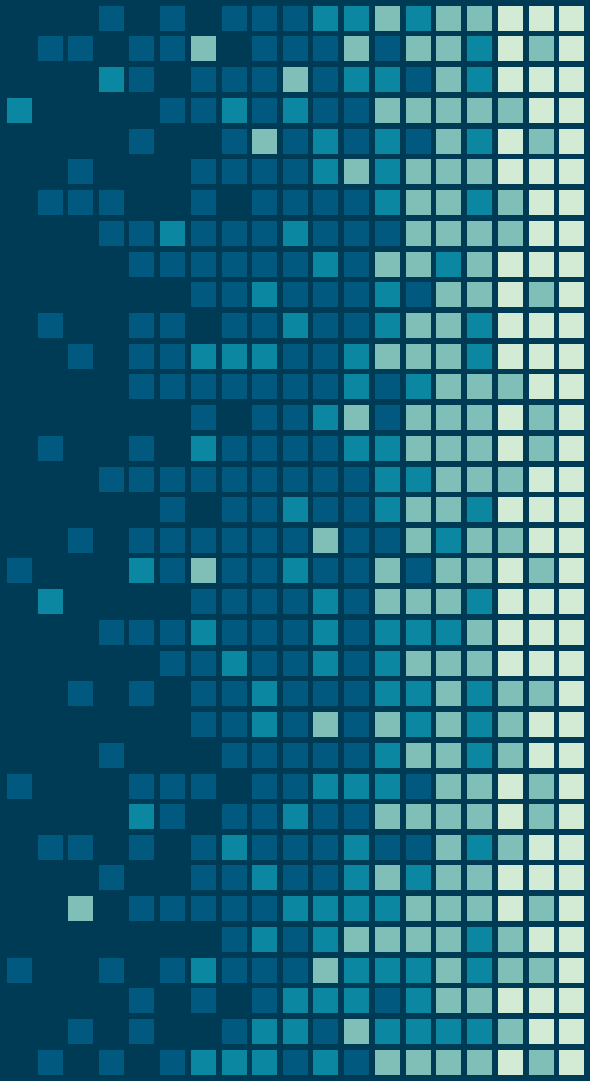
User and Operating-System Interface

- **Command Interpreters**. Algunos sistemas operativos poseen un command interpreter directo en el Kernel, otros como Windows tienen una aplicación externa para la función.
- Cuando se tienen múltiples interpretes a elegir, se llegan a conocer como **Shells**.
- Su función básica es: leer y ejecutar la instrucción dada por el usuario.
- Se pueden configurar de dos maneras: el código se encuentra dentro del interprete o bien, llama al código por separado.
- Ventaja de hacer secuencias y ciclos programados.

```
rm file.txt
```

- **Graphical User Interfaces**. También conocido como GUI.
- Se caracteriza por utilizar la metáfora del escritorio.
- 1970s at Xerox PARC
- Es usual que no todas las funcionalidades de línea de commandos sean codificadas en el entorno gráfico. (mas que todo en Linux).
- En Linux existen varias interfaces entre las cuales están K Desktop Environment (o KDE) y GNOME.

System Calls



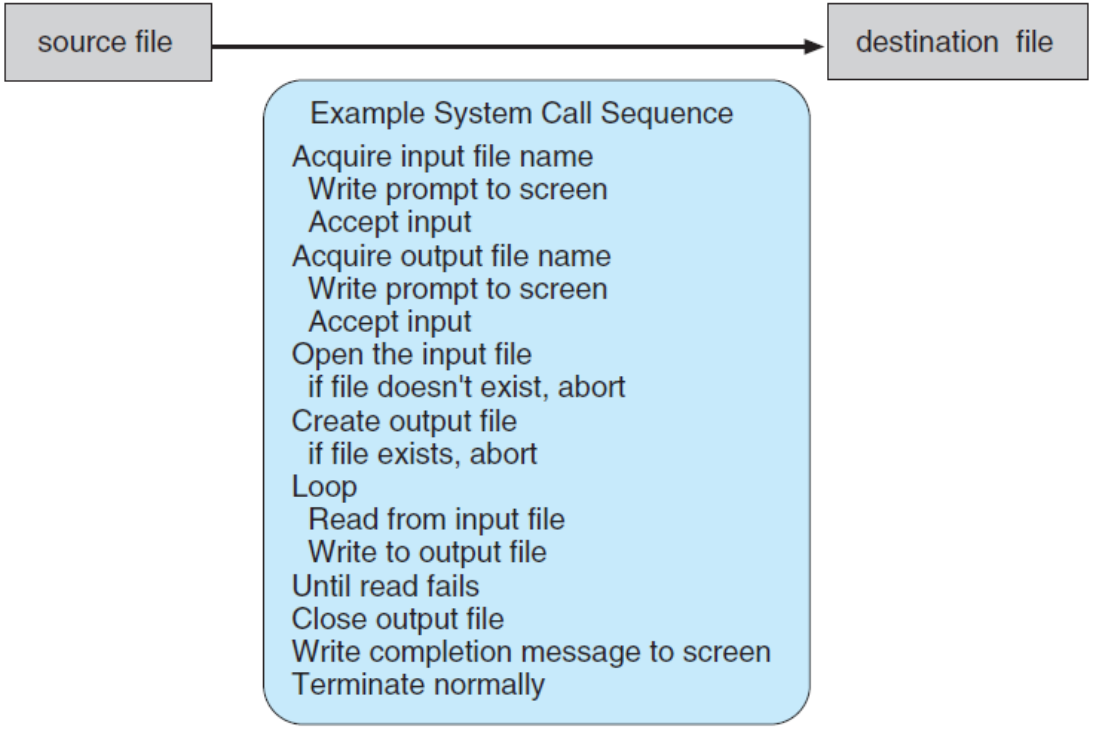


Figure 2.5 Example of how system calls are used.

- **System Calls**. Proveen una interfaz para utilizar los servicios de hardware en el sistema operativo.
- Escritas en C o en C++
- Al programador se le da un Application Programming Interface (API) . El API brinda un set de instrucciones definidas para ser usadas y a la vez estas instrucciones usan System Calls para funcionar.
- En Windows, se usa la función: `CreateProcess()`
- La cual hace uso de la system call: `NTCreateProcess()`

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)
```

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 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.

- Por qué entonces se usan APIs y no llamados directos a la System Calls?

- **Portabilidad.** Al utilizar un API, se permite utilizar la aplicación en otros entornos que posean la misma API. Ejemplo aplicaciones que funcionan gracias al Java Virtual Env. (JVE). Esto ya depende de la arquitectura de software usado.

- Que pasa al compilar un programa? Funciona solamente con el Run-Time Support System?

- **System Call Interface.** Intercepta llamadas en el aplicativo y hace las llamadas a System Calls correctas. Generalmente se tiene una tabla con índices de relación entre System Calls y la función en el API. Es esta interfaz quien controla el resultado de la System Call devuelto por el kernel en sí.

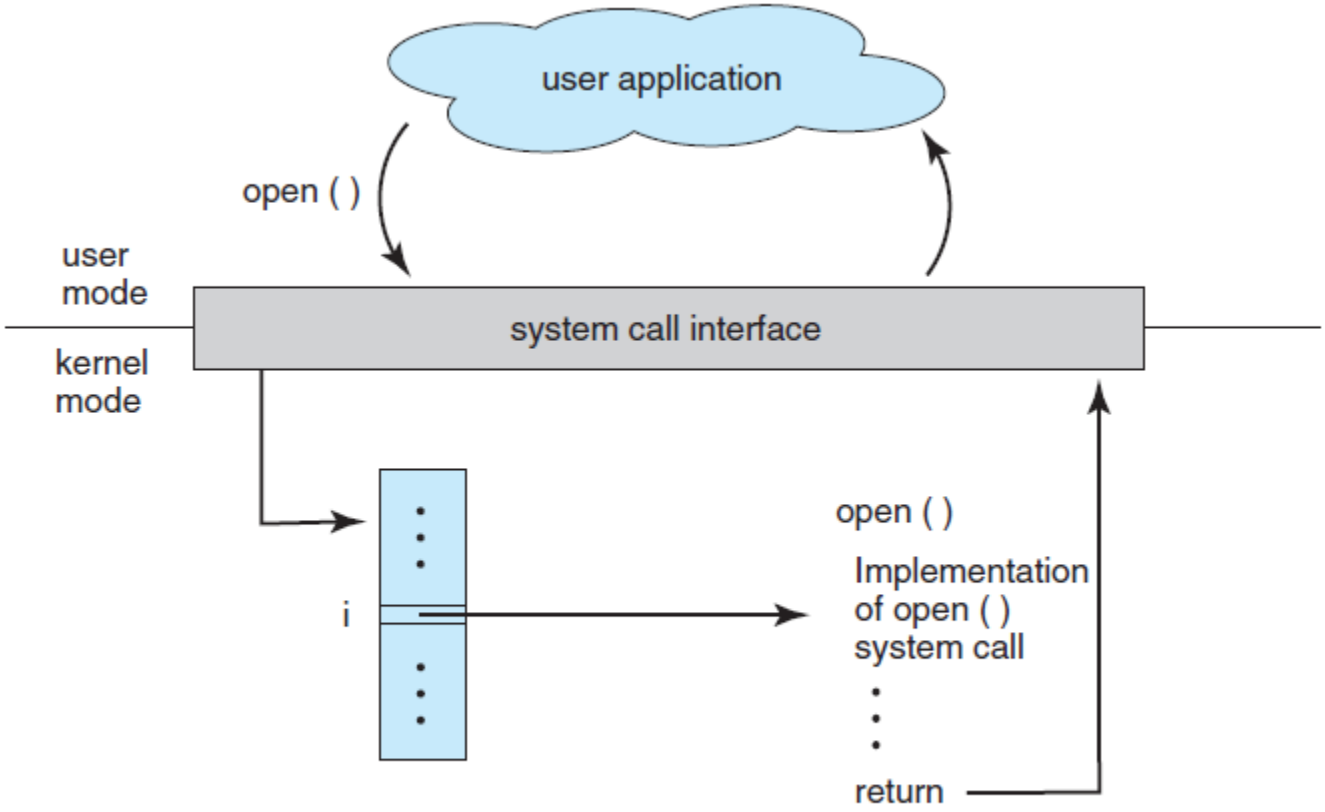


Figure 2.6 The handling of a user application invoking the `open()` system call.

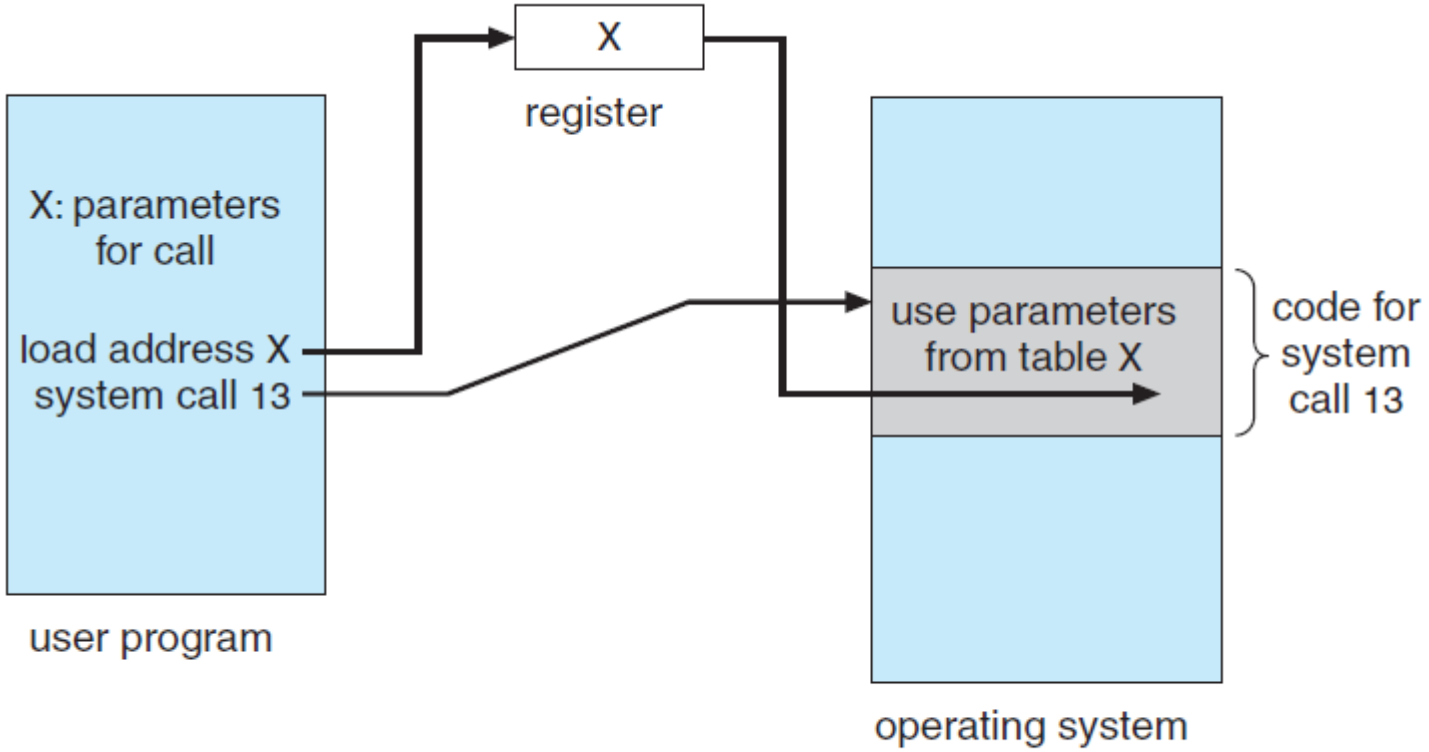


Figure 2.7 Passing of parameters as a table.