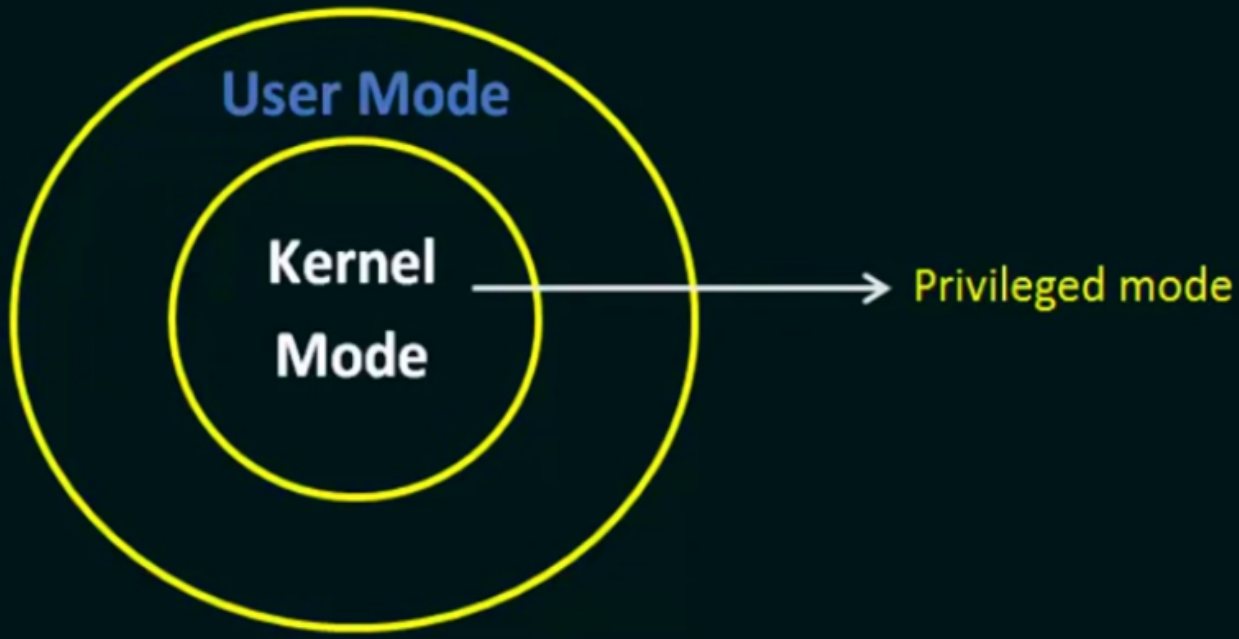
System calls provide an interface to the services made available by an Operating System.



- System call is the programmatic way in which a computer program requests a service from the kernel of the operating system.
- These calls are generally available as routines written in C and C++.

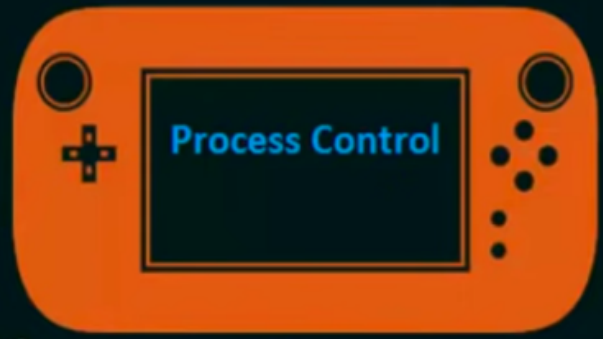
Types of System Calls

System calls can be grouped roughly into five major categories:

1. Process Control
2. File Manipulation
3. Device Management
4. Information Maintenance
5. Communications

1. Process Control

- end, abort
- load, execute
- create process, terminate process
- get process attributes, set process attributes
- wait for time
- wait event, signal event
- allocate and free memory



2. File Manipulation

- create file, delete file
- open, close
- read, write, reposition
- get file attributes, set file attributes

3. Device Manipulation

- request device, release device
- read, write, reposition
- get device attributes, set device attributes
- logically attach or detach devices

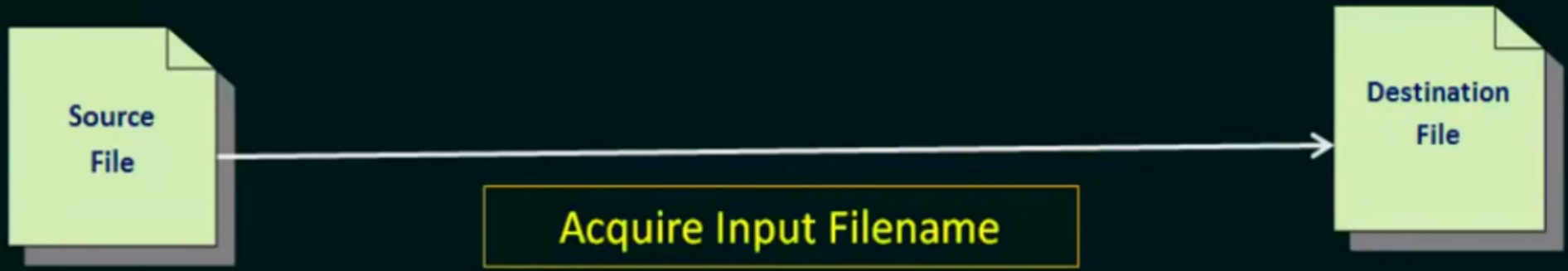
4. Information Maintenance

- 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

5. Communications

- create, delete communication connection
- send, receive messages
- transfer status information
- attach or detach remote devices

Example of a System Call sequence for writing a simple program to read data from one file and copy them to another file:



Acquire Input Filename

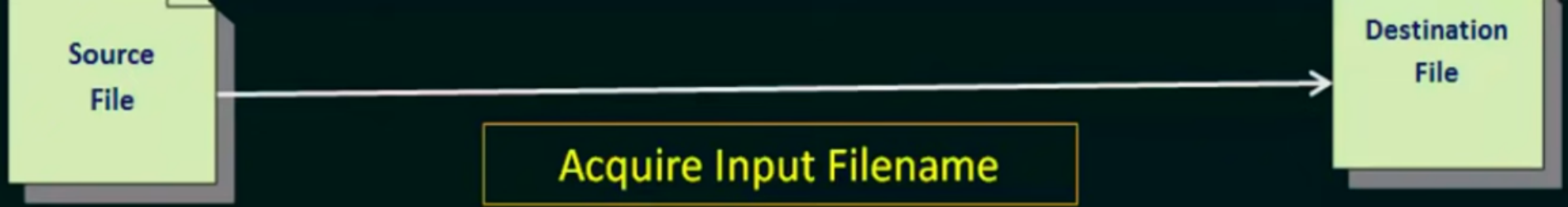
Write prompt to screen

Accept Input

Acquire Output Filename

Write prompt to screen

Accept Input



Acquire Input Filename

Write prompt to screen

Accept Input

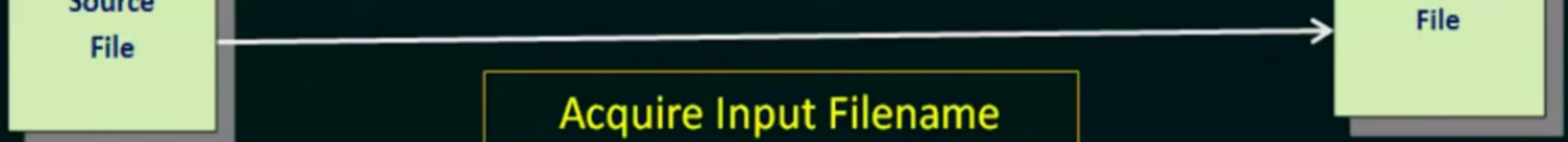
Acquire Output Filename

Write prompt to screen

Accept Input

Open Input File

If File doesn't exist, ABORT



Acquire Input Filename

Write prompt to screen

Accept Input

Acquire Output Filename

Write prompt to screen

Accept Input

Open Input File

If File doesn't exist, ABORT

Create Output File

If File exists, ABORT

Write prompt to screen

Accept Input

Open 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

example.c (~/) - gedit

1:11 AM

Open

Save

```c
#include<unistd.h>
#include<stdio.h>
#include<fcntl.h>

int main()
{
int fd;
char buffer[80];
char msg[50] = "hello ifocus institute";
fd = open ("check.txt", O_RDWR);
printf("fd = %d",fd);
if (fd != -1)
{
printf("\n check.txt opened with read write access\n");
write(fd, msg , sizeof(msg));
lseek(fd,0,SEEK_SET);
read(fd, buffer, sizeof(msg));
printf("\n %s was written to my file\n", buffer);
close (fd);
}

return 0;
}
```

C Tab Width: 8 Ln 18, Col 25 INS