Types of System Calls

System calls can be grouped roughly into five major categories.

- ✓ **Process control**
- ✓ **File management**
- ✓ **Device management**
- ✓ **Information maintenance**
- ✓ **Communications**

The following figure summarizes the types of system calls normally provided by an OS.

- 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,

7 points

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

4

- Device management
  - 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

4

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

4

Figure 2.8 Types of system calls.

Process Control

A running program needs to be able to stop execution either normally (end)or abnormally(abort). 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, often a dump of memory is taken and an error message generated. The dump is written to the disk and may be examined by a debugger.

Debugger is a system program designed to aid the programmer in finding and correcting bugs- to determine the cause of the problem. Under either normal or abnormal circumstances, the OS must transfer control to the invoking command interpreter. The CI then reads the next command. In GUI system, a popup window will alert the user to the error and ask for guidance.

A process or job executing one program may want to *load* and *execute* another program. This allows the command interpreter to execute the program. When the newly loaded program terminates , the control returns to the existing program. If both programs continue concurrently, we have created a new job or process to be multiprogrammed. (*create process).* If we create new job or process, we should be able to control its execution. The control requires the ability to determine and reset the attributes of a job or process. *(get process attribute, set process attribute).* We may want to terminate a job or process *(terminate process).* Having created new jobs or processes, we may need to wait for them to finish their execution. ***(wait time, wait event, signal event)***

File Management

Some common system calls are *create*, *delete*, *read*, *write*, *reposition*, or *close*. We first need to be able to ***create*** and ***delete*** files. Once the file is created, we need to open it and to use it. Then we may need to read, write or reposition it. Finally we need to close the file, indicating that we are no longer using it. Sometimes it is necessary to access the file attributes such as file name, file type, protection codes, accounting information and so on. To determine the file attributes – *get* and *set* file attribute are required. Many times the OS provides an API to make these system calls.

Device Management

Process usually require several resources to execute such as main memory, disk drives, access to files and so on. If these resources are available, they will be granted and control returned to the user process. Otherwise the process will have to wait until sufficient resources are available. These resources are also thought of as devices. Some are physical, such as a video card, and others are abstract, such as a file. User programs *request* the device, and when finished they *release* the device. Once the device has been requested, we can *read*, *write*, and *reposition* the device.

Information Management

Some system calls exist purely for transferring information between the user program and the operating system. An example of this is *time*, or *date*. Some other system calls may return information about the system, such as number of current users, the version number of current OS, the amount of free memory, disk space and so on. The OS also keeps information about all its processes and provides system calls to report this information.

Communication

There are two models of interprocess communication, the message-passing model and the shared memory model. In message passing model, the communicating process exchange messages with one another to transfer information. Message-passing uses a common mailbox to pass messages between processes. Before communication can take place, a connection must be opened. In Shared memory model, processes use *shared memory create* and *shared memory attach* system calls to create and gain access to regions of memory owned by other processes. The two processes exchange information by reading and writing in the shared data.