SAMUEL KIPCHUMBA ROP

AS261/1370/2014

MURANG’A UNIVERSITY COLLEGE

**IMPORTANCE OF OPERATING SYSTEM**

1. **Memory Management:** Operating System also Manages the Memory of the Computer System means Provide the Memory to the Process and Also Deallocate the Memory from the Process. And also defines that if a Process gets completed then this will deallocate the Memory from the Processes.
2. **Process Management :** **The Operating System also Treats the Process Management means all the Processes those are given by the user or the Process those are System ‘s own Process are Handled by the Operating System** . The Operating System will create the Priorities foe the user and also start or Stops the Execution of the Process and Also Makes the Child Process after dividing the Large Processes into the Small Processes.
3. **Mastermind:** Operating System also performs Many Functions and for those Reasons we can say that Operating System is a Mastermind. It provides Booting without an Operating System and Provides Facility to increase the Logical Memory of the Computer System by using the Physical Memory of the Computer System and also provides various Types of Formats like NTFS and FAT File Systems.
4. **Error detection:** Operating System also controls the Errors those have been Occurred into the Program and Also Provides Recovery of the System when the System gets Damaged Means When due to Some Hardware Failure , if System Doesn’t Works properly then this Recover the System and also Correct the System and also Provides us the Backup Facility. And Operating System also breaks the large program into the Smaller Programs those are also called as the threads. And execute those threads one by one.
5. **Device Management:** it managers devices via their respective drivers, it keeps control of input/ output device through a program called I/O controller.it also decides which process gets to a device and regulates the time interval. Also it allocates and deallocate device in an efficient way.
6. **File Management:** a file system consist of directories which contains files and directions to filed storage .The Os keep track of the file system i.e information ,usage, status, location etc.it also allocates and deallocates resources to this file system.
7. **Security:** it prevents unauthorized access to data and program through a technique of user password.
8. **Accounting:** keeps track of time and resource used by various job and user.
9. **Coordination between other software’s and users** − Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.
10. **Control over system performance** − Recording delays between request for a service and response from the system.

**WHY DISTRIBUTED SYTEM**

1. **Performance:**  very often a collection of processors can provide higher performance and better price/performance ratio than a centralized computer.
2. **Reliability**: if some of the machines crash, the system can survive.   
   Incremental growth: as requirements on processing power grow, new machines can be added incrementally.
3. **Sharing of data/resources**: shared data is essential to many applications (banking, computer supported cooperative work, reservation systems); other resources can be also shared (e.g. expensive printers).

**GOALS OF DISTRIBUTED SYSTEM**

1. **Transparency** is the concealment from the users and the application programmers of the fact that the processes and resources of a distributed system are physically distributed across multiple computers A transparent system is perceived as a whole rather than as a collection of independent components
2. **Scalability:** It is common to start any new project on a small system. If the system is successful, we will probably add more work to it over time. This means we will need more storage capacity, more network bandwidth, and more computing power. System manufacturers would be delighted if, each time we needed more capacity and power, we bought a new (larger, more expensive) computer (and threw away the old one). But
3. **Flexibility:** We may start building and testing all the parts of a new service on a notebook or desktop, but later we may decide that we need to run different parts on different computers, or a single part on multiple computers. If the components of our service interact with one-another through network protocols, it will likely be very easy to change the deployment model (which services run on which computers). Distributed systems tend to be very flexible in this respect.
4. **Openness :** An **open distributed system** is a system that offers services according to **published** standards that describe the syntax and semantics of those services– E.g., Internet is an open system as the specifications of Internet protocols are published in RFCs Services in distributed systems are generally specified through **interfaces***,* which are often described in an **Interface Definition Language** (IDL)– Interface definitions written in an IDL specify the **syntax** of the services (i.e., the names of the functions that area available, the types of the parameters, return values, and possible exceptions that can be raised)

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

int main(int argc, char \*argv[])

{

int fd;

if(2 != argc)

{

printf("\n Usage : \n");

return 1;

}

errno = 0;

/\* Through O\_CREAT flag the open() system call

creates a file if it does not exist at the

specified path.The third argument represents

the access permissions for the file \*/

fd = open(argv[1],O\_RDONLY|O\_CREAT,S\_IRWXU);

if(-1 == fd)

{

printf("\n open() failed with error [%s]\n",strerror(errno));

return 1;

}

else

{

printf("\n Open() Successful, NAME\_MAX = %d\n",NAME\_MAX);

/\* open() succeeded, now one can do read operations

on the file opened since we opened it in read-only

mode. Also once done with processing, the file needs

to be closed. Closing a file can be achieved using

close() function. \*/

}

return 0;

}