(infinite loop) or can decide to give the program more time.

- Instruction to set the value of the counter should be a privileged instruction.

18/8/15

OS Services

what are the services?

How to get the services? — user point of view — system call, system programs

designer point of view — Interrupt driven prog.

OS is also called interrupt-driven prog.

WHAT ?

Types of services

(i) Program execution

parts
- Load the prog. into mem.
- Run it
- End its execution — end (normal termination), abort (abnormal termination)

(* Every prog. has a physical end & logical end. If logical end before physical end, abnormal termination. If logical end & physical end at same time, normal termination.)

(ii) I/O operations - for files & devices (read from or write to)

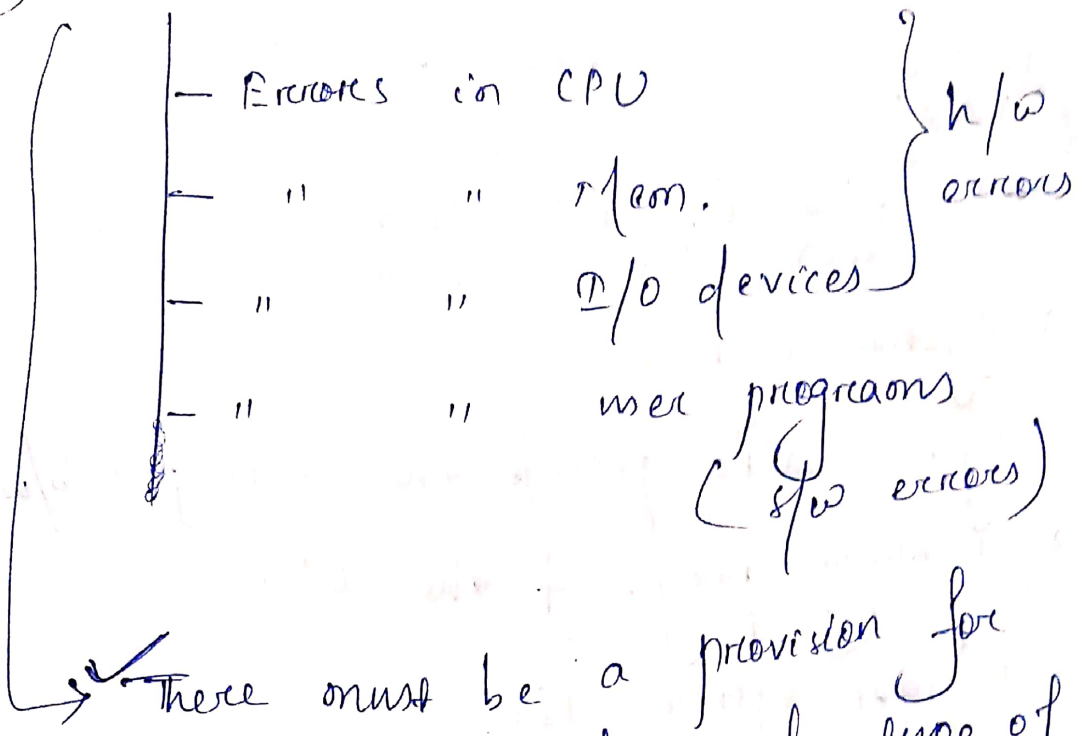
(In Unix everything is treated as a file. So I/O for files & devices are same).

(iii) File system manipulation

↳ op's are

- Create
- Open
  - implicit (when you try to read from file/create the file it's automatically opened)
  - explicit (open through a specific command)
- Read
- Write
- Reposition the file pointer
- close
- Delete

(iv) Error Detection →



There must be a provision for appropriate actions for each type of error. If no action is defined, then the system will be hanged. Or in Unix OS there is a predefined action for every ponsible error.

(v) Resource allocation — To satisfy the need of multiple users / multiple jobs running at the same time.

(vi) Accounting - (It was used previously. Now it's not needed)

- It keeps track of which user used how much & what kind of comp. resources, so that they can be charged accordingly.

(vii) Protection - (comp. from a multiuser point of view)

- One job is not allowed to interfere with others.

UNIX?

Users' view of OS services

(i) System Calls

- System calls provide the interface b/w a running prog. & the OS.
- Under system calls:

(a) Procen control or Job control

Under this the commands are:
Load, execute, end/abort, create, terminate, get/set procen attributes, wait for an Event/Time, signal the occurance of some Event/Time.

(b) File manipulation

(To create a file in DOS the internal command is copy con & external command is edit)

The commands are:
create, delete, open, close, read, write, reposition (file pointer), get/set

file attributes.

(c) Device management

(In Unix file manipulation & device management are same)

Here the op^n's are:

Request for the device, Release the device (after use), Read, Write, Reposition, Get/Set device attributes

(d) Information maintenance

Here the op^n's are:

Get/Set Date/Time, Get/Set system data

- With any of the above system calls, the control is transferred to the OS (i.e. resident monitor).
- the OS takes appropriate action for each type of call.

(ii) System programs

- System prog.s are used by OS to solve common problems & provide a convenient environment for prog. development

of execution.

(a) File manipulation

Text editor / processor is used.

(b) Programming Lang. support

like compiler, assembler, interpreter etc.

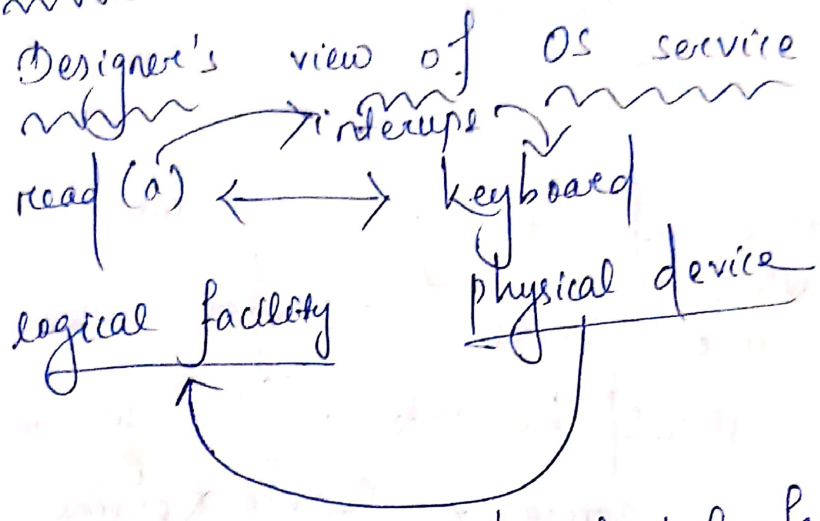
(c) Appl^n programs like Database, Statistical package etc.

(d) everything will be system calls then everything have to be loaded into memory when the system is booted. On this case the mem. will be insufficient. Accordingly the execution will be slow. There is decrease of flexibility, i.e. if we need to add some new commands then we have to replace the entire command.com file. It will be also wastage of money. Again if everything

will be system prog. then it will take more time for execution as the frequently used commands will be loaded from disk to mem. frequently.)

20/8/15

Designer's view of OS service



designer designes the logical facility of the physical devices which are going to be used by the user.

- The OS designer looks at the physical resources & devices & converts them into logical facilities to be provided to the users.
- OSs are interrupt driven programs or OSs " event-driven prog.s.
- Events are signaled by occurance of

interrupts.

- In case of interrupts the control is transferred to the OS. The OS then preserves the state of the CPU, (Process Control Block (PCB) stores the info. about the current process) determines which type of interrupt has occured.
- For each type of interrupt, separate segments of code in the OS determines what action should be taken. (The segment of code is Interrupt Service Routine (ISR)).

Types of interrupts

(1) System calls-

(a) Normal termination- Terminates the currently running prog., transfers control the Command Interpreter (CI)

(b) Abnormal termination- In case of

errors, terminates the currently running prog. after a dump of mem. with an error message & then transfers control to CI. → (In windows & but in Unix Core dump)

(Log file contains all types of semantic/runtime errors (or OS). Compiler takes care of the syntax error.)

(c) Status request (for information) - info. on date & time, attributes of files & jobs, amount of mem. space available etc. Here info. is provided & then control is transferred to the running prog.

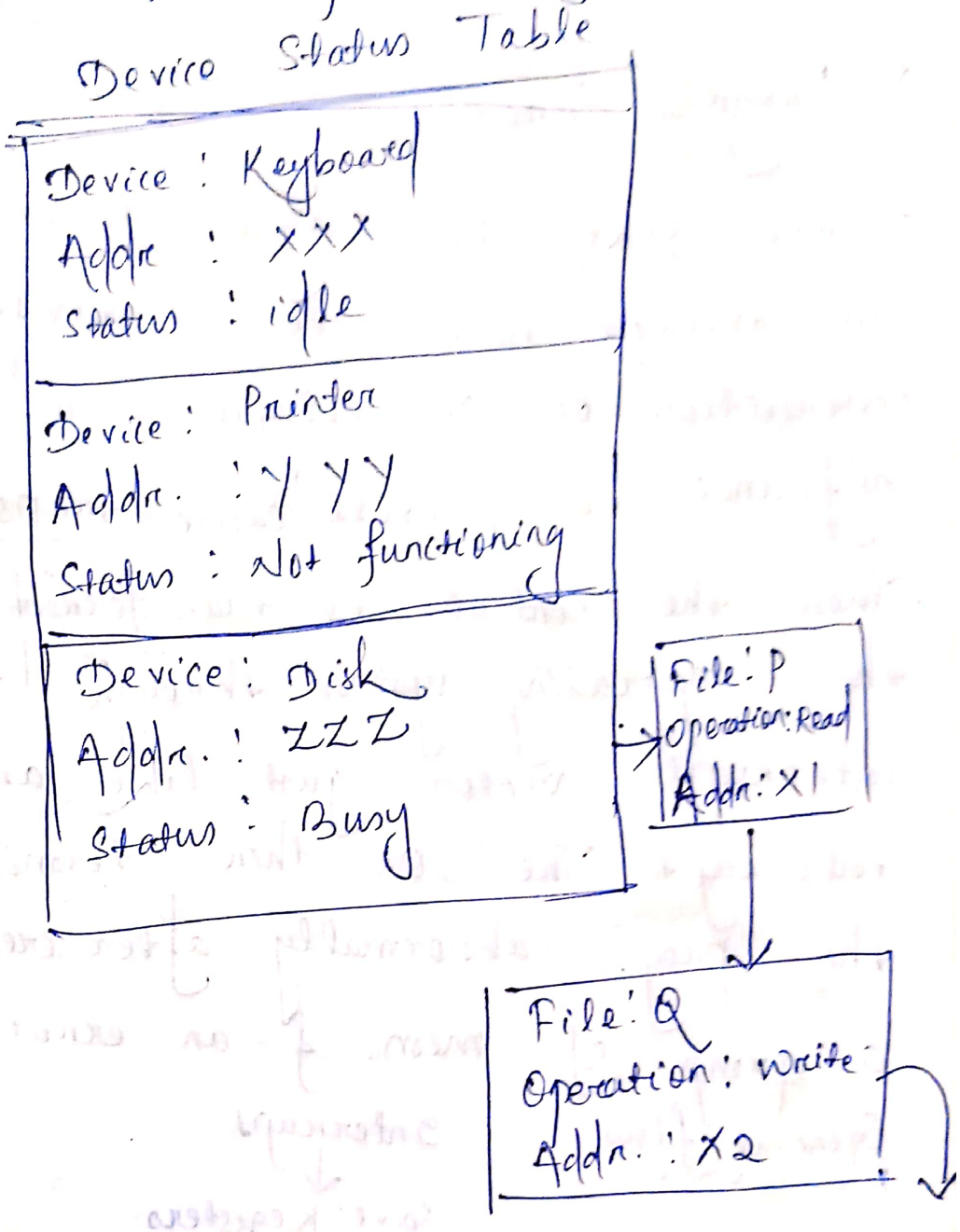
(d) Resource request - Resources like more mem., CD, Disk, acces to files etc. If the resources are available they are granted & then control is returned to the user program otherwise the prog. will have to wait.

(e) I/O request - Request for read & write op^n's

(2) I/O device interrupt

- An I/O device will interrupt when it has finished an I/O request.
- Once the I/O is started, the system can wait until it's complete or it can return to the user prog. without waiting for simultaneous I/O of many jobs.
- For simultaneous I/O of many jobs, the OS has to keep track of many I/O requests at the same time. For this purpose it maintains a table called Device Status Table, which contains an entry for each I/O device. Each entry indicates the type of the device, its physical address & its current state (i.e. whether it's idle, busy or not functioning). If the device is busy with a request, the type of the request with other parameters are stored in the table. If several requests are made

to the same device, then there will be a chain of waiting requests (which is a linked list (queue implemented by a linked list)).



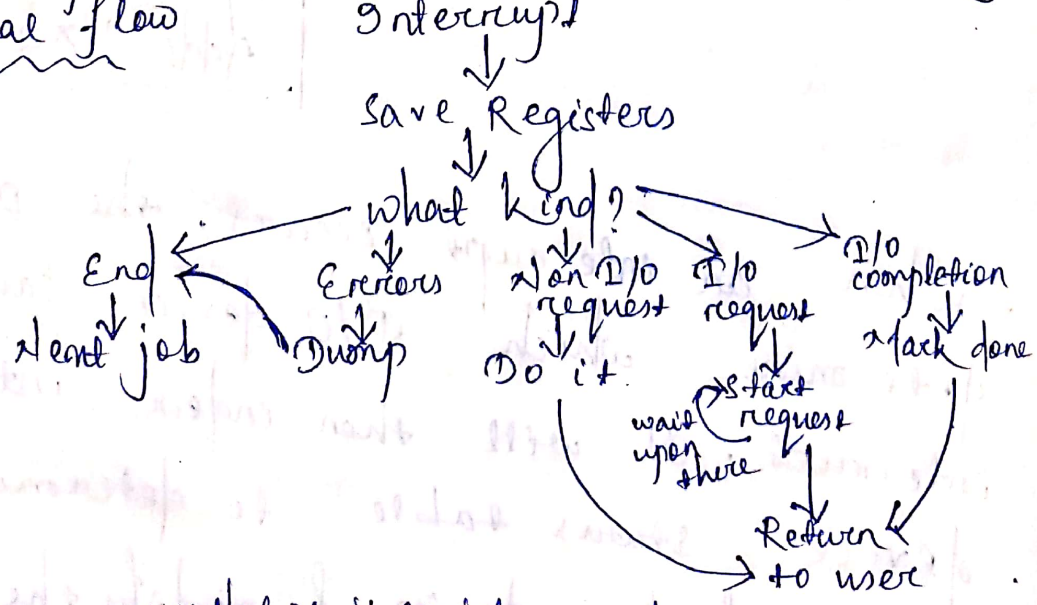
- when an interrupt occures, the OS will determine which I/O device cause the interrupt. It will then index into the device status table to determine the status of the device & modify the

table entry to reflect the occurance of the interrupt. Finally the control returns from the I/O interrupt.

(3) Program Errors

- Errors, like an illegal instruction, an attempt to execute a privileged instruction or an illegal memory reference etc., will cause traps. Then the control is transferred to the operating system through the interrupt vector just like an interrupt. The OS then terminates the prog. abnormally after creating a dump of mem. & an error message.

General flow



(This diagram says that OS is an interrupt driven prog.)
Non I/O request → request for info.