Assignment No-6

D D M M Y Y Y Y

19UCS122

Q1. Write short note on:

(a) Source of interrupts

→ The processor receives interrupts from two sources:

(1) External (Hardware)-generated interrupts.

(i) External interrupts are received through pins on the processor or through the local APIC

(ii) Maskable Hardware interrupts

Any external interrupts that is delivered to the processor by means of the INTR pin or through the local APIC is called a maskable hardware interrupt

(2) Software generated interrupts.

(i) The INT n instruction permits interrupts to be generated from within software by supplying an interrupt vector number as an operand.

(ii) Example: the INT 35 instruction forces an implicit call to the interrupt handler for interrupt 35.

(iii) Interrupts generated in software with the INT n instruction cannot be masked by the IF flag in the EFLAGS register.

(b) Interrupts and Exception

→ (i) Interrupts occur at random times during the execution of a program, in response to signals from hardware.

(ii) Software can also generates interrupts by executing the INT n instruction

(iii) Exceptions occur when the processor detects an error condition while executing an instruction, such as division by zero.

(iv) When an interrupt is received or an exception is detected, the currently running procedure or task is suspended while the processor executes an interrupt or exception handler.

(v) When execution of the handler is complete, the processor resumes execution of the interrupted procedure or task.

Q2. Write &0386 assembly Language program to display return values of system calls related to process.

→.

```
.section .data
.section .bss
    .lcomm pid, 4
    .lcomm uid, 4
    .lcomm gid, 4

.section .text
.globl _start
_start:
    movl $20, %eax
    int $0x80
    movl %eax, pid
    movl $24, %eax
    int $0x80
    movl %eax, uid
    movl $47, %eax
    int $0x80
    movl %eax, gid
end:
    movl $1, %eax
    movl $0, %ebx
    int $0x80
```

19UCS122

Q 3. Write short Note on

(a) Linux System Call interface

→ (1) Linux support multiple processes running simultaneously

(2) At any time, memory images of multiple processes exist simultaneously with the same physical memory.

(3) Multiple resources (IO devices) are used by a process in mutually exclusive manner with other processes

(4) A resource is used only by one process at a time and other processes wait for this process to release the resource before they can use it.

(5) To protect a process by other processes, Linux do not permit to do direct IO i.e. The OS does not permit OS code to be executed by a process in user mode.

(6) A process makes a call to a function within an OS to perform such operations. Such functionality within OS is called system call.

(7) IA32 processor provide instruction for handling operating system calls.

(8) INT N N is an immediate constant ranging from 0 to 255.

(9) In Linux, Instruction int $0x80 is used for making system call to the operating system

19UCS122

(b) System call Identification:

→ (1) Linux provide a mechanism for entering into system call by using it int 0x80 instruction.
(2) After executing int 0x80, control of program transfers to a predefined location within os. This code is called as system call handler.
(3) System call handler handle various system call such as read, write, open etc. as per the system call identification process.
(4) Each system call in Linux is given a unique number for identification.
(5) In Linux register EAX is used to pass the system call number.

(c) Return values from System calls.

→ (1) Return value of system calls is implimented similar to function calls.
(2) System call return their values in eax register
(3) There are two kinds of values
(A) Non negative (B) Negative.
(4) All non negative values returned by the system call represents successful execution of system call, whereas negative values represents error conditions.
(5) for write system call, it returns the size of the string written to the file descriptor.

System call value	System call	Description
20	getpid	Retrieves the process ID
24	getpid	Retrieves the user ID
47	getpid	Retrieves the group ID

19UCS122

(d) Starting a process in Linux.

→ (1) In Linux, a program is executed by creating a process and loading a program in newly created process

(2) A fork system call is used to create a process and a program is loaded using execve system call.

(3) When a program is loaded, it run from memory location whose address is specified in executable file.

(4) By default starting address of the program is indicated by label _start.

Q4. List and explain different system calls related to process management.

→

System Call Name	System Call value	Description
Fork		create a child process
Execve		Execute a program
Exit		Terminate the current process
Nice		change the process priority
kill		send a signal to kill a process.

Q5. Write &0386 assembly language program to pass input parameters for write system call. Also display that message "Computer science" on terminal screen.

19UCS122

```
.section .data
output:
        .ascii "computer Science.\n"
output_end:
.eqn len, output_end - output
.section .text
.globl _start
_start:
        movl $4 , %.eax
        movl $1 , %.ebx
        movl $output, %.ecx
        movl $len, %.edx
        int $0x80
        movl $1, %.eax
        movl $0, %.ebx
        int $0x80
```