Sotware Interrupt/ Exceptions/ Process state transitions

An interrupt is the automatic transfer of software execution in response to a hardware event that is asynchronous with the current software execution.

This hardware event is called a **trigger**.

The hardware event can either be a busy to ready transition in an external I/O device or an internal event (like bus fault, memory fault, or a periodic timer).

After receiving an interrupt, the processor completes execution of the current instruction, then pauses the current process

a) The processor will then execute one of the kernel's interrupt handling functions

b) The interrupt handler determines how the system should respond

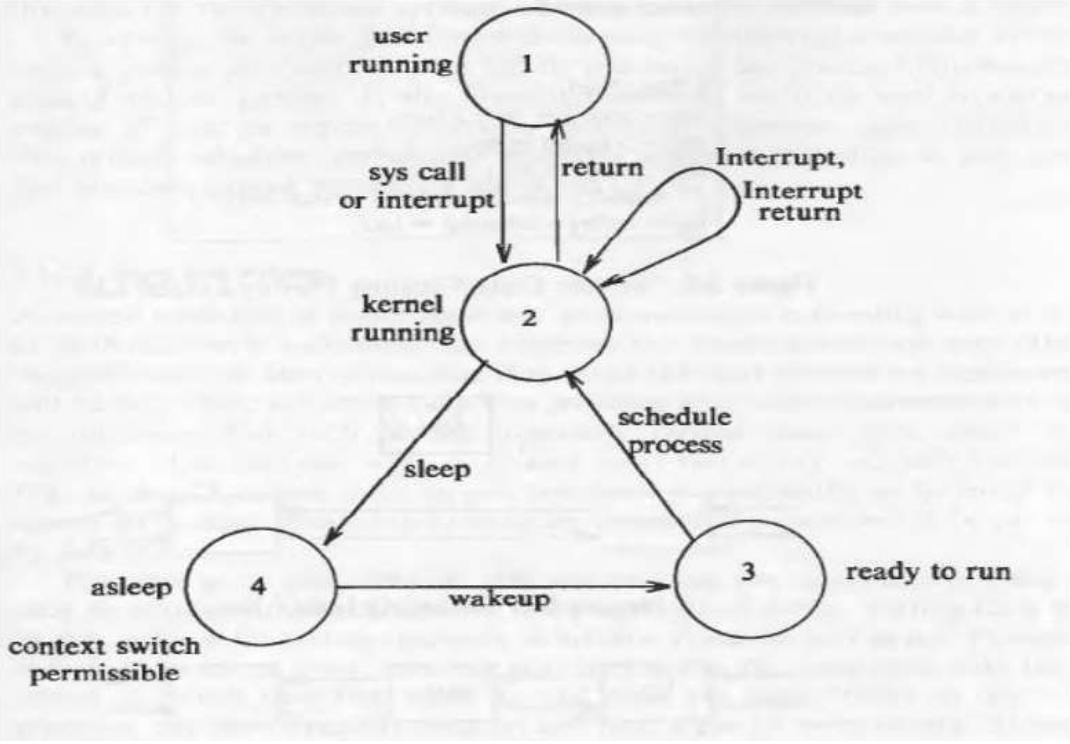


Figure { SEQ Figure * ARABIC }: Process state transition

Explain process state transition

A **software interrupt**, also called an exception, is an interrupt that is caused by **{** HYPERLINK "http://www.linfo.org/software.html" **}**, usually by a **{** HYPERLINK "http://www.linfo.org/program.html" **}** in **{** HYPERLINK "http://www.linfo.org/user_mode.html" **}**.

An interrupt is a signal to the **{** HYPERLINK "http://www.linfo.org/kernel.html" **}** (i.e., the core of the **{** HYPERLINK "http://www.linfo.org/operating_systems_list.html" **}**) that an event has occurred, and this results in changes in the sequence of instructions that is executed by the CPU (central processing unit).

The lifetime of a process can be divided into a set of states, each with certain characteristics that describe the process.

1. The process is currently executing in user mode.

2. The process is currently executing in kernel mode.

3. The process is not executing, but it is ready to run as soon as the scheduler chooses it. Many processes may be in this state, and the scheduling algorithm determines which one will execute next.

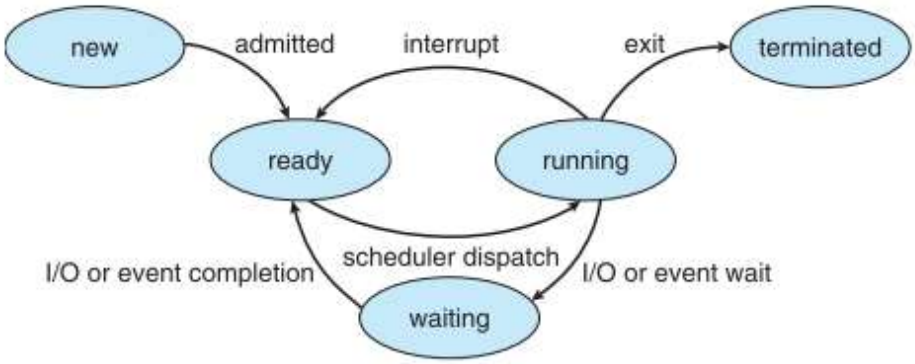
4. The process is sleeping. A process puts itself to sleep when it can no longer continue executing, such as when it is waiting for I/O to complete.

Because a processor can execute only one process at a time, at most one process may be in states 1 and 2.

The process states described above give a static view of a process, but processes move continuously between the states according to well-defined rules. A state transition diagram is a directed graph whose nodes represent the states a process can enter and whose edges represent the events that cause a process to move from one state to another. State transitions are legal between two states if there exists an edge from the first state to the second.

Figure 1. shows the state transition diagram for the process states defined above.

The kernel allows a context switch only when a process moves from the state "kernel running" to the state "asleep in memory." Processes running in kernel mode cannot be preempted by other processes; therefore the kernel is sometimes said to be non-preemptive, although the system does preempt processes that are in user mode. The kernel maintains consistency of its data structures because it is non-preemptive, thereby solving the mutual exclusion problem — making sure that critical sections of code are executed by at most one process at a time.



Difference between Interrupts and Exceptions

Interrupts and exceptions both alter the program flow. The difference between the two is that interrupts are used to handle external events (serial ports, keyboard) and exceptions are used to handle instruction faults, (division by zero, undefined opcode).

Interrupts are handled by the processor after finishing the current instruction. If it finds a signal on its interrupt pin, it will look up the address of the interrupt handler in the interrupt table and pass that routine control. After returning from the interrupt handler routine, it will resume program execution at the instruction after the interrupted instruction.

Exceptions on the other hand are divided into three kinds.

These are Faults, Traps and Aborts.

Faults are detected and serviced by the processor before the faulting instructions.

Traps are serviced after the instruction causing the trap. User defined interrupts go into this category and can be said to be traps.

Aborts are used only to signal severe system problems, when operation is no longer possible