Process Description and Control

Introduction

- The fundamental task of any operating system is process management.
- OS must allocate resources to processes, enable sharing of information, protect resources, and enable synchronization among processes.
- In many modern OS the problems of process management is compounded by introduction of threads.

Topics

- **Requirement of process**
- **Process states**
- **Creation, termination and suspension**
- **Five State Model**
- **Process Control Block (PCB)**
- **Process control**

What is a process?

- A process is simply a program in execution: an instance of a program execution.
- Unit of work individually schedulable by an operating system.
- OS keeps track of all the active processes and allocates system resources to them according to policies devised to meet design performance objectives.
- To meet process requirements OS must maintain many data structures efficiently.
- The process abstraction is a fundamental OS means for management of concurrent program execution. Example: instances of process co-existing.

Major requirements

- OS must interleave the execution of a number of processes to maximize processor use while providing reasonable response time.
- OS must allocate resources to processes in conformance with a specific policy. Example: (i) higher priority, (ii) avoid deadlock.
- Support user creation of processes and IPC both of which may aid in the structuring of applications.

Process creation

Four common events that lead to a process creation are:

1) When a new batch-job is presented for execution.

2) When an interactive user logs in.

3) When OS needs to perform an operation (usually IO) on behalf of a user process, concurrently with that process.

4) To exploit parallelism an user process can spawn a number of processes.

==> concept of parent and child processes.

Termination of a process

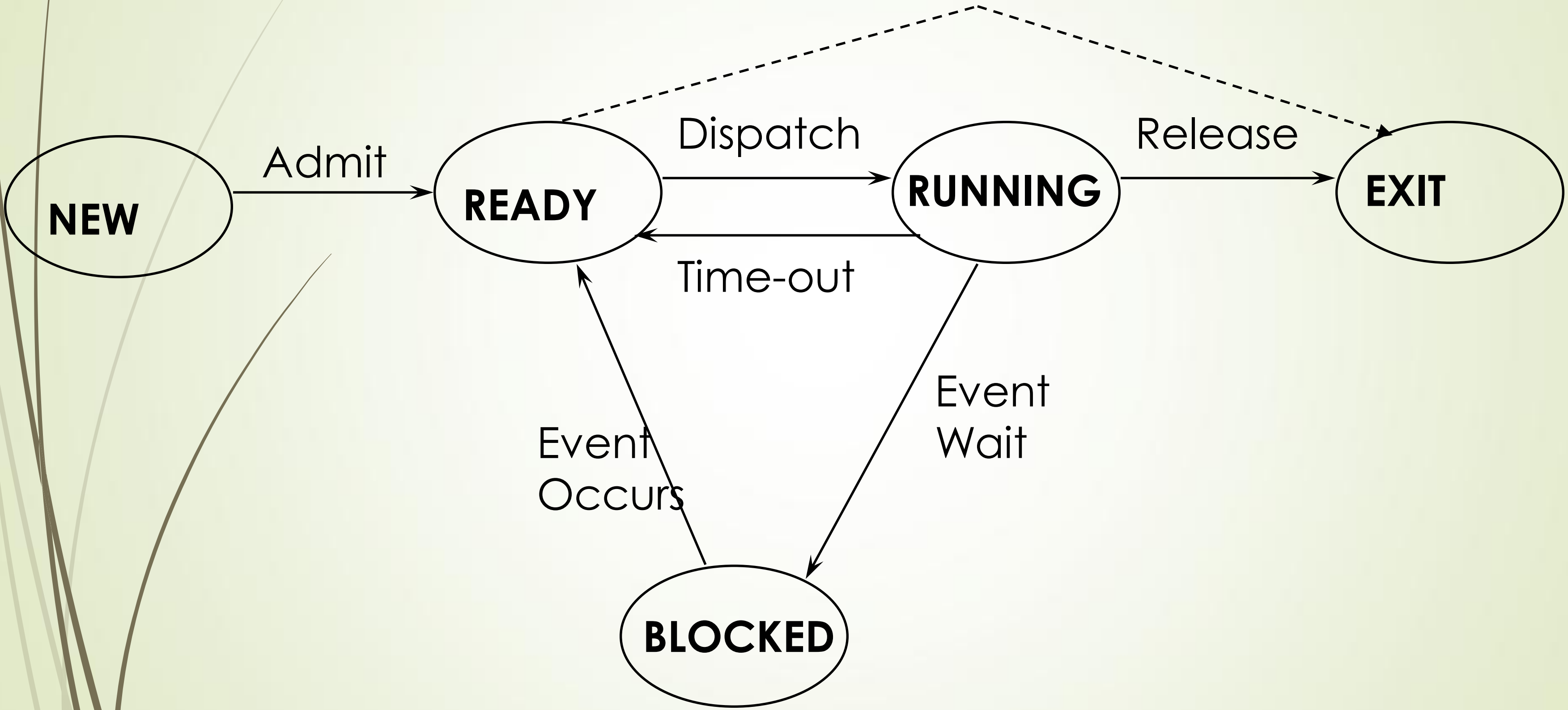
- Normal completion, time limit exceeded, memory unavailable
- Bounds violation, protection error, arithmetic error, invalid instruction
- IO failure, Operator intervention, parent termination, parent request
- A number of other conditions are possible.
- **Segmentation fault** : usually happens when you try write/read into/from a non-existent array/structure/object component. Or access a pointer to a dynamic data before creating it. (new etc.)
- **Bus error:** Related to function call and return. You have messed up the stack where the return address or parameters are stored.

A five-state process model

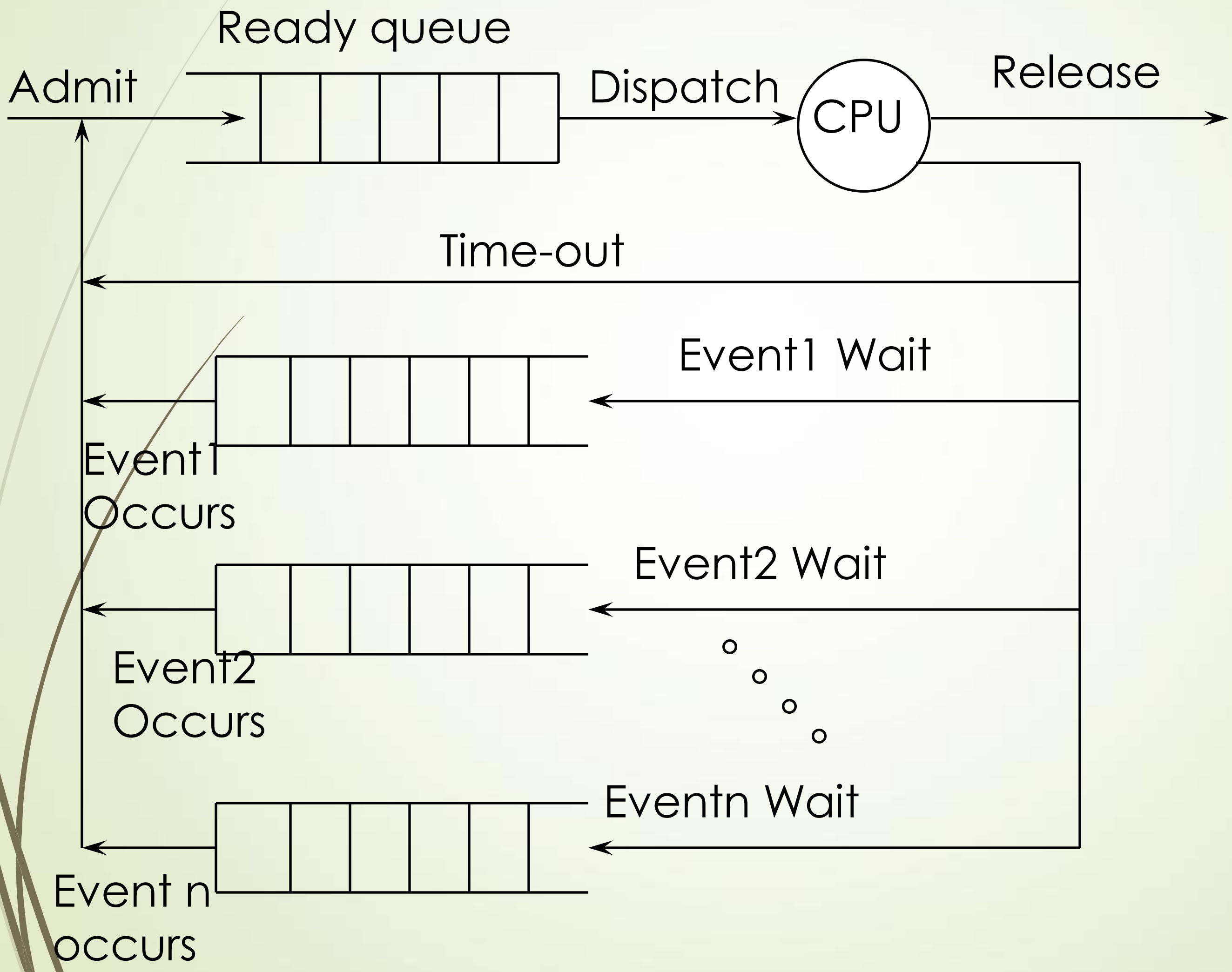
Five states: New, Ready, Running, Blocked, Exit

- **New** : A process has been created but has not yet been admitted to the pool of executable processes.
- **Ready** : Processes that are prepared to run if given an opportunity. That is, they are not waiting on anything except the CPU availability.
- **Running**: The process that is currently being executed. (Assume single processor for simplicity.)
- **Blocked** : A process that cannot execute until a specified event such as an IO completion occurs.
- **Exit**: A process that has been released by OS either after normal termination or after abnormal termination (error).

State Transition Diagram



Queuing model



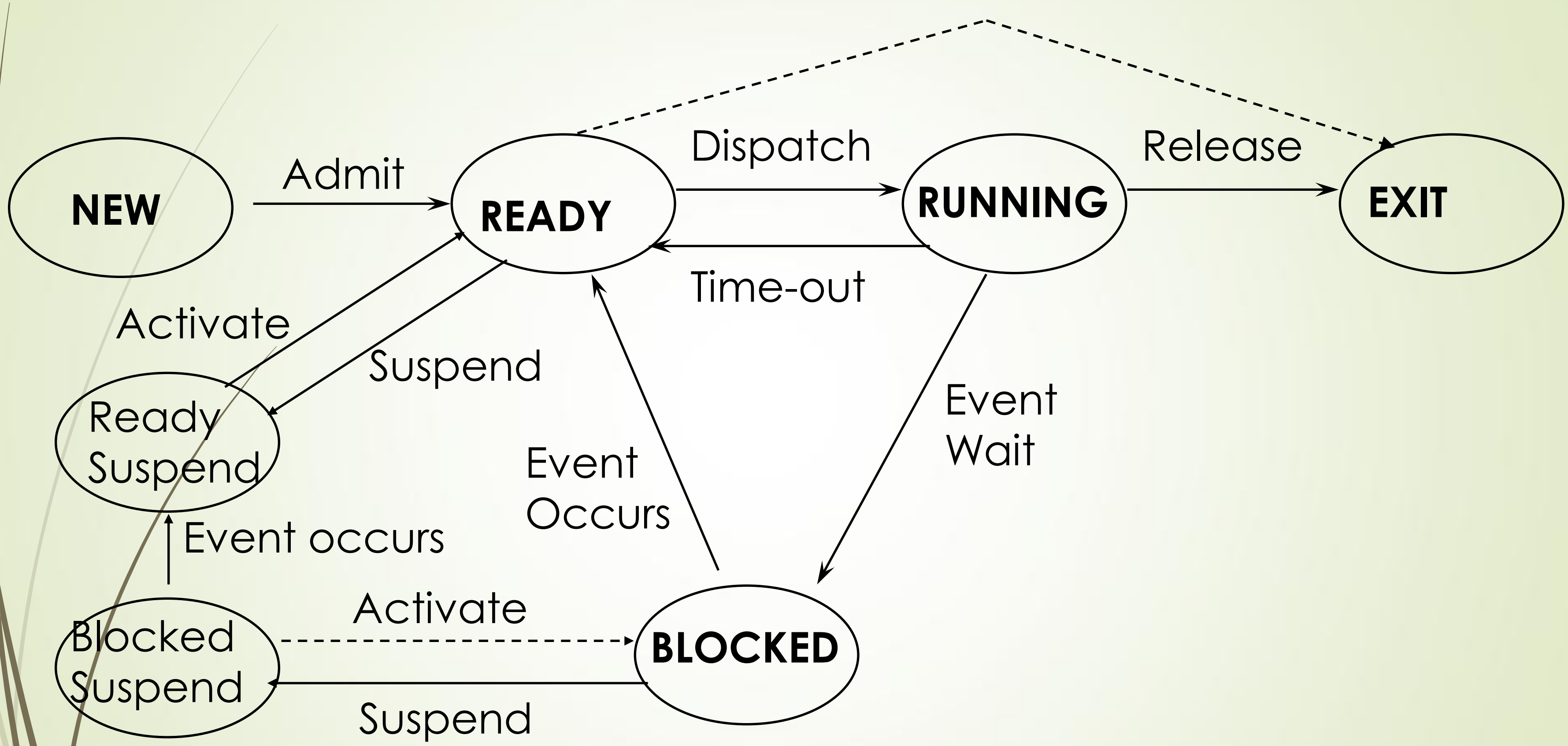
Process suspension

- Many OS are built around (Ready, Running, Blocked) states. But there is one more state that may aid in the operation of an OS - **suspended** state.
- When none of the processes occupying the main memory is in a Ready state, OS swaps one of the blocked processes out onto to the Suspend queue.
- When a Suspended process is ready to run it moves into “Ready, Suspend” queue. Thus we have two more state: Blocked_Suspend, Ready_Suspend.

Process suspension (contd.)

- Blocked_suspend : The process is in the secondary memory and awaiting an event.
- Ready_suspend : The process is in the secondary memory but is available for execution as soon as it is loaded into the main memory.
- Observe on what condition does a state transition take place? What are the possible state transitions?

State Transition Diagram



Process description

- OS constructs and maintains tables of information about each entity that it is managing : memory tables, IO tables, file tables, process tables.
- **Process control block:** Associated with each process are a number of attributes used by OS for process control. This collection is known as **PCB.**
- Process image: Collection of program, data, stack, and PCB together is known as **Process image**

Process control block

- Contains three categories of information:

1) Process identification

2) Process state information

3) Process control information

- **Process identification**:
  1) numeric identifier for the process (pid)
  2) identifier of the parent (ppid)
  3) user identifier (uid) - id of the usr responsible for the process.

Process control block (contd.)

Process state information:

1) User visible registers
2) Control and status registers : PC, IR, PSW, interrupt related bits, execution mode.
3) Stack pointers

Process control information:

1) Scheduling and state information : Process state, priority, scheduling-related info., event awaited.
2) Data structuring : pointers to other processes (PCBs): belong to the same queue, parent of process, child of process or some other relationship.
3) Interprocess comm: Various flags, signals, messages may be maintained in PCBs.

OS Functions related to Processes

- Process management: Process creation, termination, scheduling, dispatching, switching, synchronization, IPC support, management of PCBs
- Memory management: Allocation of address space to processes, swapping, page and segment management.
- IO management: Buffer management, allocation of IO channels and devices to processes.
- Support functions: Interrupt handling, accounting, monitoring.

Modes of execution

- Two modes : user mode and a privileged mode called the kernel mode.
- Why? It is necessary to protect the OS and key OS tables such as PCBs from interference by user programs.
- In the kernel mode, the software has complete control of the processor and all its hardware.
- When a user makes a system call or when an interrupt transfers control to a system routine, an instruction to change mode is executed. This mode change will result in an error unless permitted by OS.

Process Interruption

- Two kinds of process interruptions: **interrupt** and **trap**.
- **Interrupt:** Caused by some event external to and asynchronous to the currently running process, such as completion of IO.
- **Trap** : Error or exception condition generated within the currently running process. Ex: illegal access to a file, arithmetic exception.
- (supervisor call) : explicit interruption.

THANKS