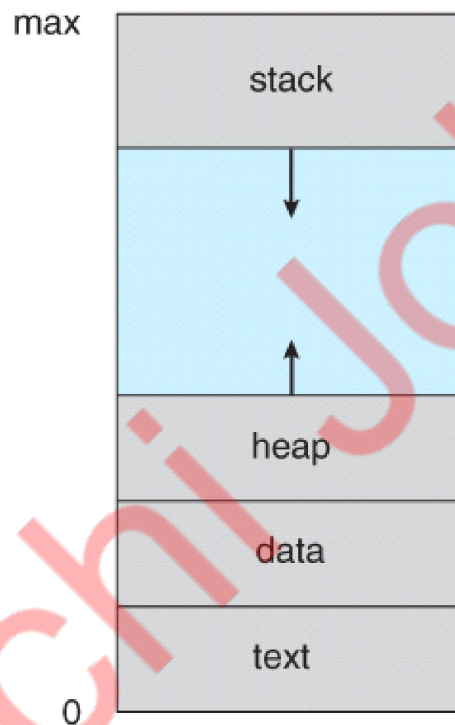


## Process Concept

- A process is an instance of a program in execution.
- Batch systems work in terms of "jobs". Many modern process concepts are still expressed in terms of jobs, ( e.g. job scheduling ), and the two terms are often used interchangeably.

## The Process

- Process in memory is divided into four sections as shown in Figure below:



- The text section comprises the compiled program code, read in from non-volatile storage when the program is launched.
- The data section stores global and static variables, allocated and initialized prior to executing main.
- The heap is used for dynamic memory allocation, and is managed via calls to new, delete, malloc, free, etc.
- The stack is used for local variables. Space on the stack is reserved for local variables when they are declared, and the space is freed up when the variables go out of scope.

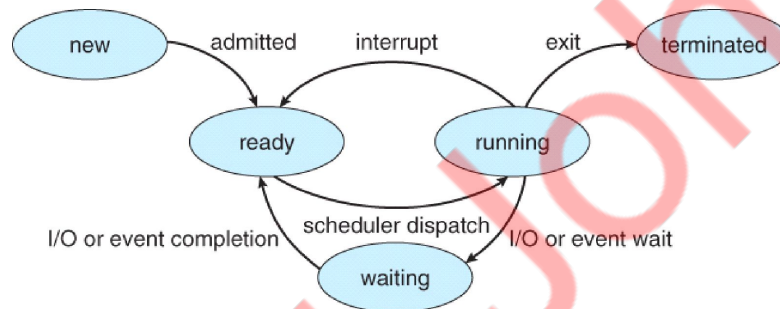
**Note** that the stack is also used for function return values, and the exact mechanisms of stack management may be language specific.

**Note** that the stack and the heap start at opposite ends of the process's free space and grow towards each other. If they should ever meet, then either a stack overflow error will occur, or else a call to new or malloc will fail due to insufficient memory available.

- When processes are swapped out of memory and later restored, additional information must also be stored and restored. Key among them are the program counter and the value of all program registers.

### Process State

- Processes may be in one of 5 states, as shown in Figure below.



- **New** - The process is in the stage of being created.
- **Ready** - The process has all the resources available that it needs to run, but the CPU is not currently working on this process's instructions.
- **Running** - The CPU is working on this process's instructions.
- **Waiting** - The process cannot run at the moment, because it is waiting for some resource to become available or for some event to occur.

**Example** - the process may be waiting for keyboard input, disk access request, inter-process messages, a timer to go off, or a child process to finish.

- **Terminated** - The process has completed.

### Process Scheduling

- The two main objectives of the process scheduling system are to keep the CPU busy at all times and to deliver "acceptable" response times for all programs, particularly for interactive ones.

- The process scheduler must meet these objectives by implementing suitable policies for swapping processes in and out of the CPU.

**Note:** These objectives can be conflicting. In particular, every time the system steps in to swap processes it takes up time on the CPU to do so, which is thereby "lost" from doing any useful productive work.

### ***Scheduling Queues***

- All processes are stored in the **job queue**.
- Processes in the Ready state are placed in the **ready queue**.
- Processes waiting for a device to become available or to deliver data are placed in **device queues**. There is generally a separate device queue for each device.
- Other queues may also be created and used as needed.