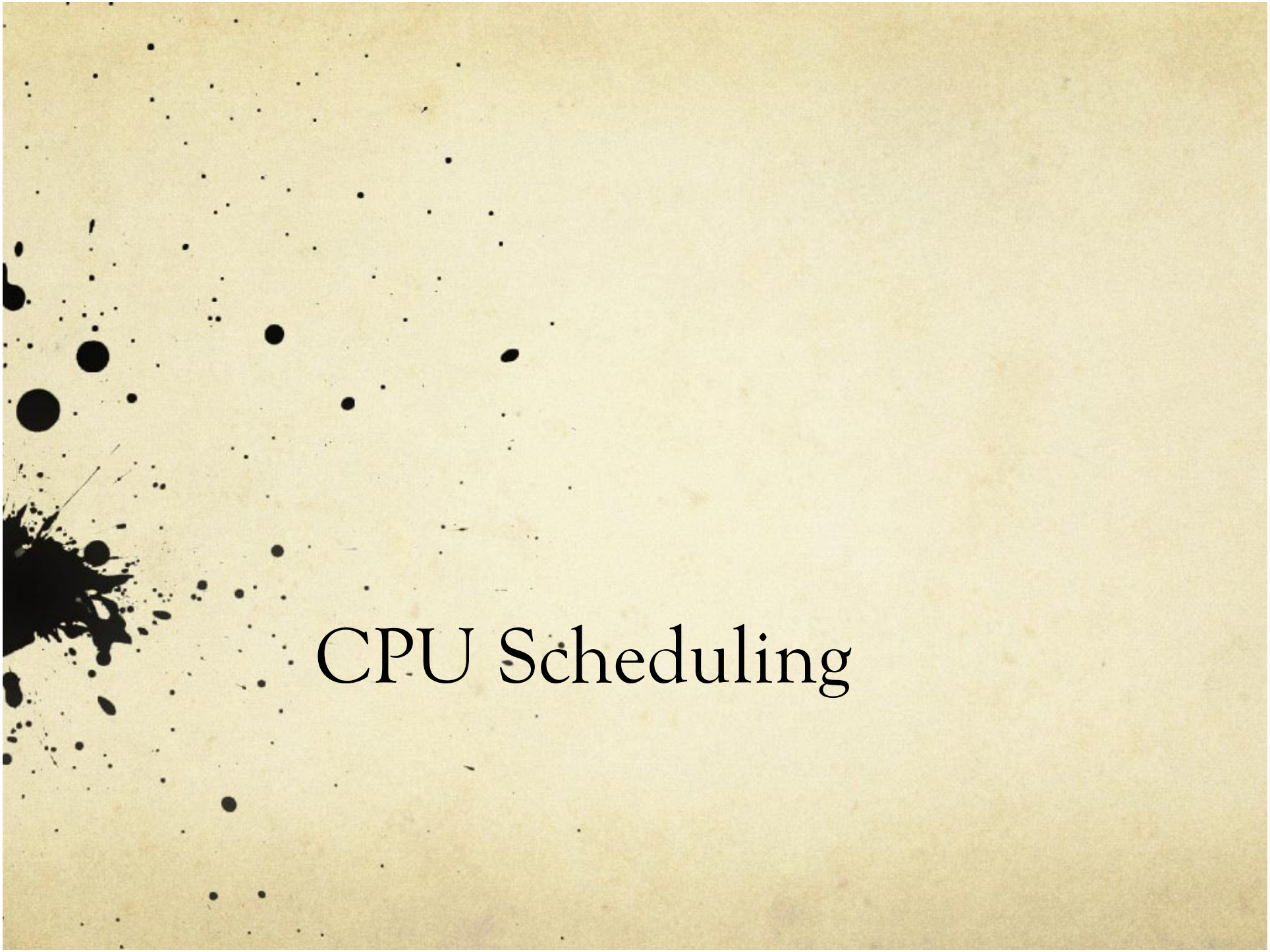

CPU Scheduling

We have used the term 'scheduling' informally several times. Let us now formally understand what it means and how it is done.

What is CPU scheduling?

- When system has multi-programming
  - Several jobs may be *ready* for execution
  - Ready jobs *compete* for CPU use

- Need to decide *which* job to run
  - Part of OS that performs scheduling → *scheduler*
  - Algorithm used to make decision → *scheduling algorithm*

- Our focus will be on *policies* for scheduling

What kind of policy should be used?

- Depends on system or environment
  - *Batch* system
    - Set of jobs submitted to system
    - Can optimize policy for *overall* system performance
  - *Interactive* system
    - Users directly waiting for results of jobs or commands
    - Policies must optimize *user-perceived* performance
  - *Real-time* system
    - Jobs have *deadlines* by which they must complete
    - Policies must ensure *predictability*

Objectives of scheduling algorithm

- General objectives (for all types of systems)
  - *Fairness*
    - I.e., make sure each job gets fair share of CPU
  - *Efficiency* & *balance*
    - Try to keep system busy as much as possible
  - *Policy enforcement*
    - Ensure that all chosen policies appropriately carried out

Objectives of scheduling algorithm

- Batch systems
  - *Throughput*
    - Maximize # of jobs executed in given time interval (e.g., one hour)
  - *Turnaround time*
    - Minimize time between submission & completion of jobs
  - *CPU utilization*
    - Keep CPU utilized at all times

Objectives of scheduling algorithm

- Interactive systems
  - *Response time*
    - Minimize time b/w issue & completion of interactive jobs
  - *Proportionality*
    - Users
      - Accept that *long* jobs may have *long* response times
      - Expect *short* jobs to have *short* response times
    - Maintain this perception

Objectives of scheduling algorithm

- Real-time systems
  - *Predictability*
  - *Deadline*
    - Ensure that job deadlines are met
      - Deadline miss could
        - Be catastrophic (*hard-real-time* system)
        - Degrade Quality of Service or result in data loss (*soft-real-time* system)
  - *Priorities*
    - Priorities need to be considered in scheduling decisions

When should scheduling be done?

- *Depends* on type of environment
- In general, scheduling can be done when
  - *New process* is created
  - Process *terminates*
  - Process gets *blocked*
    - Waiting for I/O
    - Waiting for lock on shared resource
  - *Interrupt* occurs
    - I/O completion
    - Timer expiration

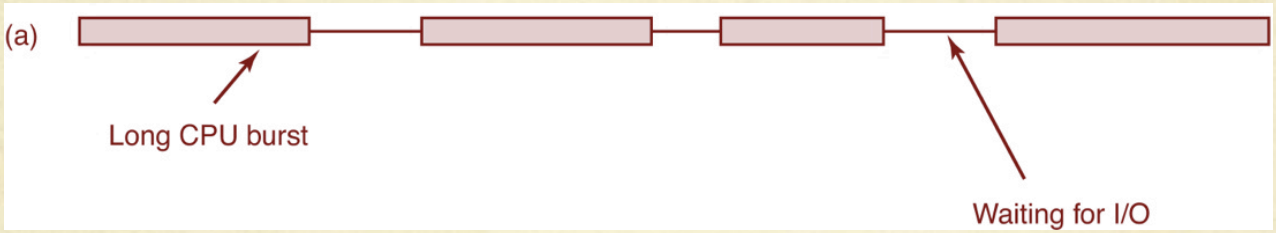
Types of scheduling

- *Preemptive* scheduling
  - Executing job can be interrupted & other job scheduled

- *Non-preemptive* scheduling
  - Once job is chosen for execution, it continues until it *completes*, *blocks* or voluntarily *yields* CPU

Job characteristics

- Compute boundedness
  - *Long* bursts of CPU execution before blocking for I/O



- I/O boundedness
  - *Short* bursts of CPU execution before blocking for I/O