Scheduling Policies
Real-time scheduling

- Real-time processes have timing constraints
  - Timing constraints are translated into deadlines or rate requirements

Two dominant real-time scheduling policies

"Rate-Monotonic" scheduling
Priority = 1/rate (the "period")

"Deadline" scheduling
Priority = release time + deadline

Example: Digital video playout

/* Main processing loop */
loop
  data = read(network)
  video_frame = decompress(data)
  write(frame_buffer, video_frame)
end loop

Timing constraint: Execute loop once every 33 ms.
Scheduling Policies
Real-time scheduling example

- Consider scheduling audio and video playout processes
  - The audio process processes 1 audio sample every 20 ms
  - The video process processes 1 video frame every 33 ms

Deadline scheduling: \[ \text{priority} = \text{release time} + \text{deadline} \]
\[ \text{deadline} = \text{inter-arrival time (in this example)} \]
Scheduling Policies

Real-time scheduling example

- Consider scheduling audio and video playout processes
  - The audio process processes 1 audio sample every 20 ms
  - The video process processes 1 video frame every 33 ms

Deadline scheduling:  
\[ \text{priority} = \text{release time} + \text{deadline} \]
\[ \text{deadline} = \text{inter-arrival time (in this example)} \]
Consider scheduling audio and video playout processes

- The audio process processes 1 audio sample every 20 ms
- The video process processes 1 video frame every 33 ms

Deadline scheduling: priority = release time + deadline
deadline = inter-arrival time (in this example)
Scheduling Policies
Real-time scheduling example

- Consider scheduling audio and video playout processes
  - The audio process processes 1 audio sample every 20 ms
  - The video process processes 1 video frame every 33 ms

Deadline scheduling:  
\[ \text{priority} = \text{release time} + \text{deadline} \]  
\[ \text{deadline} = \text{inter-arrival time (in this example)} \]
Consider scheduling audio and video playout processes

- The audio process processes 1 audio sample every 20 ms
- The video process processes 1 video frame every 33 ms

Deadline scheduling: 
\[ \text{priority} = \text{release time} + \text{deadline} \]
\[ \text{deadline} = \text{inter-arrival time} \text{ (in this example)} \]
Consider scheduling audio and video playout processes
- The audio process processes 1 audio sample every 20 ms
- The video process processes 1 video frame every 33 ms

Deadline scheduling: $\text{priority} = \text{release time} + \text{deadline}$
$\text{deadline} = \text{inter-arrival time (in this example)}$
Scheduling Policies

Real-time scheduling example

- Consider scheduling audio and video playout processes
  - The audio process processes 1 audio sample every 20 ms
  - The video process processes 1 video frame every 33 ms

Deadline scheduling: \[\text{priority} = \text{release time} + \text{deadline}\]

\[\text{deadline} = \text{inter-arrival time (in this example)}\]
Consider scheduling audio and video playout processes:

- The audio process processes 1 audio sample every 20 ms.
- The video process processes 1 video frame every 33 ms.

Deadline scheduling: 

\[
\text{priority} = \text{release time} + \text{deadline} \\
\text{deadline} = \text{inter-arrival time} \text{ (in this example)}
\]
Scheduling Policies
Real-time scheduling example

- Consider scheduling audio and video playout processes
  - The audio process processes 1 audio sample every 20 ms
  - The video process processes 1 video frame every 33 ms

Deadline scheduling: priority = release time + deadline
deadline = inter-arrival time (in this example)
Scheduling Policies
Real-time scheduling example

Consider scheduling audio and video playout processes

- The audio process processes 1 audio sample every 20 ms
- The video process processes 1 video frame every 33 ms

Deadline scheduling: \( \text{priority} = \text{release time} + \text{deadline} \)
\( \text{deadline} = \text{inter-arrival time} \) (in this example)
**Scheduling Policies**

**Real-time scheduling example**

- Consider scheduling audio and video playout processes
  - The audio process processes 1 audio sample every 20 ms
  - The video process processes 1 video frame every 33 ms

Deadline scheduling: $\text{priority} = \text{release time} + \text{deadline}$

$\text{deadline} = \text{inter-arrival time} \ (\text{in this example})$
Scheduling Policies
Real-time scheduling example

- Consider scheduling audio and video playout processes
  - The audio process processes 1 audio sample every 20 ms
  - The video process processes 1 video frame every 33 ms

Deadline scheduling: priority = release time + deadline
deadline = inter-arrival time (in this example)
Scheduling Policies
Real-time scheduling example

Consider scheduling audio and video playout processes
- The audio process processes 1 audio sample every 20 ms
- The video process processes 1 video frame every 33 ms

Deadline scheduling: \[ \text{priority} = \text{release time} + \text{deadline} \]
\[ \text{deadline} = \text{inter-arrival time} \] (in this example)
Consider scheduling audio and video playout processes

- The audio process processes 1 audio sample every 20 ms
- The video process processes 1 video frame every 33 ms

Deadline scheduling: $priority = release\ time + deadline$

deadline = inter-arrival time (in this example)
Consider scheduling audio and video playout processes:
- The audio process processes 1 audio sample every 20 ms.
- The video process processes 1 video frame every 33 ms.

Rate-monotonic scheduling: priority $= 1 / \text{rate} = \text{period}$.
Consider scheduling audio and video playout processes:
- The audio process processes 1 audio sample every 20 ms.
- The video process processes 1 video frame every 33 ms.

Rate-monotonic scheduling: \( \text{priority} = \frac{1}{\text{rate}} = \text{period} \)
Consider scheduling audio and video playout processes
- The audio process processes 1 audio sample every 20 ms
- The video process processes 1 video frame every 33 ms

Rate-monotonic scheduling: $\text{priority} = \frac{1}{\text{rate}} = \text{period}$
Scheduling Policies

Real-time scheduling example

- Consider scheduling audio and video playout processes
  - The audio process processes 1 audio sample every 20 ms
  - The video process processes 1 video frame every 33 ms

Rate-monotonic scheduling: \( priority = 1 / rate = period \)
**Scheduling Policies**

*Real-time scheduling example*

- Consider scheduling audio and video playout processes
  - The audio process processes 1 audio sample every 20 ms
  - The video process processes 1 video frame every 33 ms

Rate-monotonic scheduling: \( priority = 1 / rate = period \)
Scheduling Policies

Real-time scheduling example

- Consider scheduling audio and video playout processes
  - The audio process processes 1 audio sample every 20 ms
  - The video process processes 1 video frame every 33 ms

Rate-monotonic scheduling: \( \text{priority} = 1 / \text{rate} = \text{period} \)
Consider scheduling audio and video playout processes
- The audio process processes 1 audio sample every 20 ms
- The video process processes 1 video frame every 33 ms

Rate-monotonic scheduling: \( \text{priority} = 1 / \text{rate} = \text{period} \)
Consider scheduling audio and video playout processes:

- The audio process processes 1 audio sample every 20 ms
- The video process processes 1 video frame every 33 ms

Rate-monotonic scheduling: $priority = 1/rate = period$
Scheduling Policies
Real-time scheduling example

Consider scheduling audio and video playout processes

- The audio process processes 1 audio sample every 20 \( ms \)
- The video process processes 1 video frame every 33 \( ms \)

Rate-monotonic scheduling:  \( \text{priority} = \frac{1}{\text{rate}} = \text{period} \)
Consider scheduling audio and video playout processes.
- The audio process processes 1 audio sample every 20 ms.
- The video process processes 1 video frame every 33 ms.

Rate-monotonic scheduling: $\text{priority} = 1 / \text{rate} = \text{period}$
Consider scheduling audio and video playout processes:

- The audio process processes 1 audio sample every 20 ms.
- The video process processes 1 video frame every 33 ms.

Rate-monotonic scheduling: \( \text{priority} = \frac{1}{\text{rate}} = \text{period} \)
**Scheduling Policies**

**Real-time scheduling example**

- Consider scheduling audio and video playout processes
  - The audio process processes 1 audio sample every 20 ms
  - The video process processes 1 video frame every 33 ms

Rate-monotonic scheduling: \( priority = 1 / rate = period \)
Consider scheduling audio and video playout processes

- The audio process processes 1 audio sample every 20 ms
- The video process processes 1 video frame every 33 ms

Rate-monotonic scheduling: \( \text{priority} = \frac{1}{\text{rate}} = \text{period} \)
Consider scheduling audio and video playout processes

- The audio process processes 1 audio sample every 20 ms
- The video process processes 1 video frame every 33 ms

Rate-monotonic scheduling: \( \textit{priority} = \frac{1}{\text{rate}} = \text{period} \)
Scheduling Policies: Summary

- FCFS:
  - Not fair, and poor average waiting times

- Priority Scheduling:
  - Not Fair and starvation possible

- Round Robin:
  - Fair, but poor average waiting times

- SJF/SRTF:
  - Not fair, but average waiting time is minimized
  - Requires accurate prediction of computation times
  - Starvation is possible

- MLF:
  - An approximation to SJF

- Real-time scheduling:
  - Needed to meet timeliness constraints
Memory Management Basics
Basic Memory Management Concepts

Address spaces

- **Physical address space** — The address space supported by the hardware
  - Starting at address 0, going to address $\text{MAX}_{\text{sys}}$

- **Logical/virtual address space** — A process’s view of its own memory
  - Starting at address 0, going to address $\text{MAX}_{\text{prog}}$

But where do addresses come from?

```
MOV r0, @0xfffffa620e
```
Basic Concepts
Address generation

- The compilation pipeline

```
prog P:
  foo():
  inc SP, x
  jmp _foo:
  foo:...
end P
```

```
P:
  push ...
  inc SP, 4
  jmp 75:
  ...
```

```
Library Routines
  push ...
  inc SP, 4
  jmp 175:
  ...
```

```
Library Routines
  push ...
  inc SP, 4
  jmp 1175:
  ...
```

Compilation → Assembly → Linking → Loading
Program P’s logical address space

Instructions

Logical Addresses

≤

CPU

MEMORY EXCEPTION

no

yes

Program P’s physical address space

Base Register

1000

≤

500

Limit Register

+ Physical Addresses

CPU

1000

1500

MAX_{sys}

0