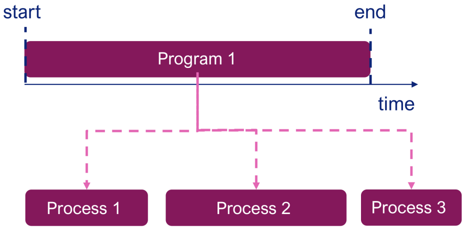
EssCS - Topic 3

Introduction to Operating Systems

Lecture 11, 12.11.2024

Nuša Zidarič

Recap: Multitasking

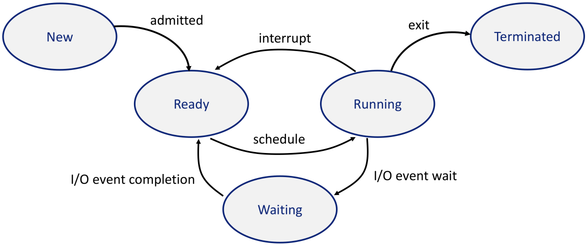


- recall: process is (a part of) a program in execution
- Program 1 as 3 processes (with different exec. times!)
- switching between processes is faster then switching between "programs"
- timesharing approach

Recap: How OS manages processes

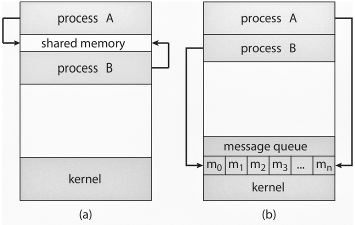
- process: insns and data
- the OS keeps a *process control block (PCB)* for each process
- PCB includes sufficient information for a process to be interrupted and resumed
  - process state (new, running, waiting, ready, terminated)
  - process number
  - PC (new insn to be executed)
  - CPU registers (depends on computer architecture)
  - memory limits (OS will allow access only if process within its limits)
  - I/O status (list of open files, allocated I/O devices, ...)
  - Accounting information (processor time, time limits, ...)
- Switching between processes is faster then switching between “programs”
- timesharing approach

Recap: How OS manages processes



- the figure above is showing possible states for a process
- the OS keeps queues for processes: ready queue, device queue (e.g. for each IO controller),. . . , and it moves the processes between queues
- some processes are more IO intensive, some are more CPU intensive, some can be both → important for maximizing resource usage

InterProcess Communication (IPC)



- Independent processes: can not affect / can not be affected by other processes
- Cooperating: they affect / can be affected by other processes

- Cooperating processes → need for communication
  - Information sharing (multiple processes need access to same file)
  - Computation speedup (divide workload and merge results)
  - Modularity (dividing system functions into separate processes)

Resource Allocation

- if resource not available, process[1] enters the waiting state
- deadlock: we have a set of processes and every process in this set is waiting for an event that can only be caused by another process in this set

- deadlock conditions - all four must be met:
  - mutual exclusion (resource non-shareable)
  - hold-and-wait (process holds a resource and waits for another resource)
  - no preemption (system can not "take-away" the resource)
  - circular wait

- resource allocation graph (RAG)
- nodes: processes P_i and resources R_i
- directed edges: request edge $P_i \rightarrow R_j$ and assignment edge $R_i \rightarrow P_i$

- how to eliminate deadlocks:
- abort all processes in the set
- abort one process at a time until deadlock eliminated

[1] also applies to threads