System-Level Programming

27 Programs and Processes

Peter Wägemann

Lehrstuhl für Informatik 4
Systemsoftware

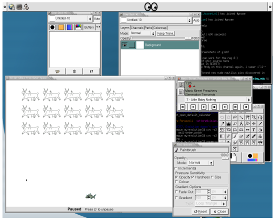
Friedrich-Alexander-Universität
Erlangen-Nürnberg (FAU)

Summer Term 2025

http://sys.cs.fau.de/lehre/ss25

Overview

- **Multiple** Programs that
- run **concurrently**,
- are **dynamically** started/stopped,
- control their environment
- via **defined I/O functions**.



Source: www.wikipedia.org

Each running program gets hardware assigned:

- CPU (time shares)
- memory (parts of the main memory)
- ...and can call operating-system–kernel functions

Definitions

Program: set of instructions

Process: running program and its data

Hint: one program can be in execution multiple times (e.g., PDF viewer)!

Processes

- Definition “process”: running program with its data
- Different point of view:

microcontroller process	UNIX-/Windows/... process
processor	time shares of the physical processor
memory	virtual memory
interrupts	signals
I/O devices	I/O operating-system functions

Processes (2)

- Multi-program operation (“multitasking”)
  - multiple processes can be executed virtually simultaneously
  - if there are less processors then there are running processes, time shares for using a processor are given to the processes: **time-sharing system**
  - the OS kernel the decides which process receives how much computing time: **scheduling**
  - the switch between processes takes place by the OS kernel: **dispatching**
  - running processes do not know at which point a subsequent process is dispatched

Process States

A process is always in one of the following states

New (or created):
: Process has been created but does not have all necessary resources to run

Ready:
: Process has all necessary resources (except CPU) and is ready for execution/running

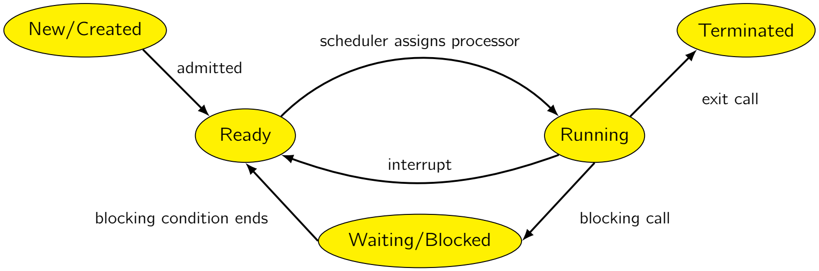
Running:
: Process is executed by a physical processor

Waiting (or blocked):
: Process waits for an event (completion of an I/O operation)

Terminated:
: Process is terminated but not all of its resources are yet freed

Process States (2)

- State diagram with transitions:

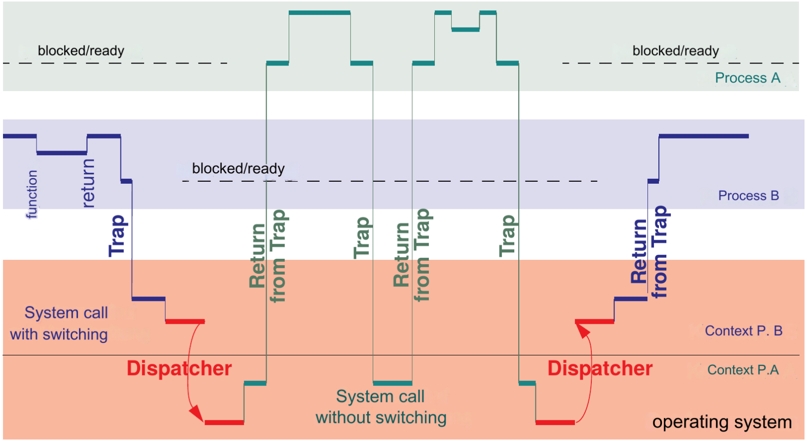


Context Switch

- Each process has a context (i.e., its state)
  - contents of processor registers
  - contents of memory areas
  - open files, current directory, ...
- When switching a process (context switch)
  - the contents of the processor registers are saved,
  - a new process is selected,
  - the execution environment for the new process is established
    - reprogramming of the MMU
    - change of the open files and current working directory, ...
  - the stored registers of the new process are loaded

Context Switch

- control flow of two processes in user mode and kernel



Process Control Block

- Process Control Block (PCB)

  Data structure of the kernel that contains all necessary data for a process.

  Example UNIX:

  - process ID (PID)
  - process state (running, ready, ...)
  - register
  - memory mapping
  - owner (UID, GID)
  - root directory, working directory
  - open files
  - ...