Operating Systems

Overview of the course

Cristian Vidrașcu

https://profs.info.uaic.ro/~vidrascu

Contents

- Objectives of the OS course
- Resources
- Overview of courses and laboratories
- Examination (ro)
- Feedback

Objectives of the OS course:

1. Students will acquire the knowledge about operating systems, regarding the techniques used for the design and the implementation of them.

2. Develop students' skills to use the Linux OS and to write parallel processing programs on Linux.

Objectives

Skills acquired:

1) Learning the basic concepts concerning the functioning of operating systems

2) Understanding the architecture of an operating system with its main components

3) Understanding the algorithms that are used by an operating system for resource management

4) Using the text interface provided by the UNIX/Linux OS and the parallel processing techniques available

5) Designing software applications that use the services provided by an UNIX/Linux OS

Resources

- Webpage of the OS course:

  https://profs.info.uaic.ro/~vidrascu/OS/

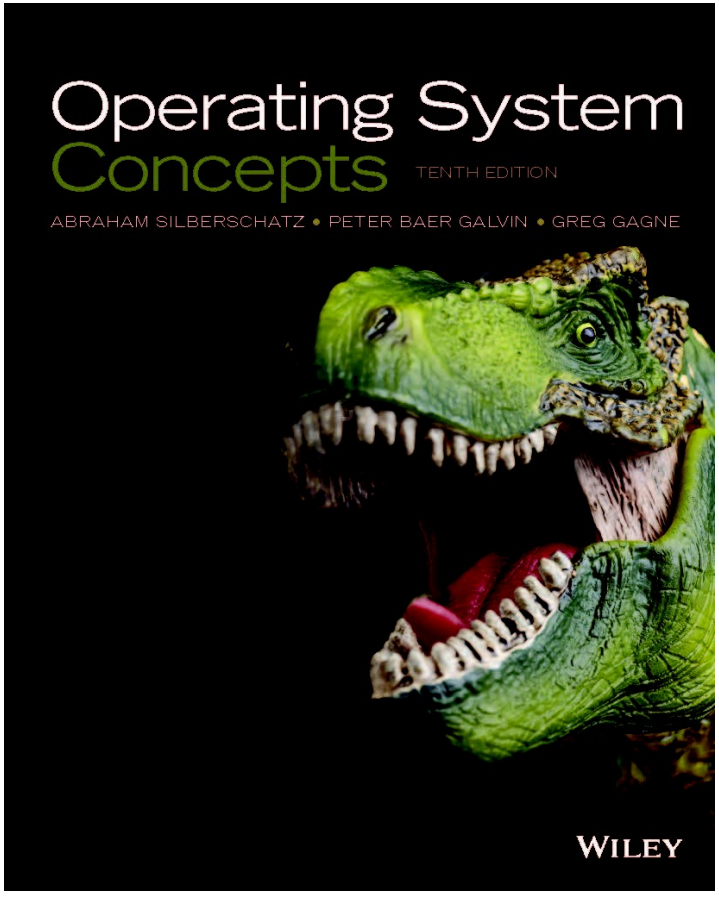
- Books:

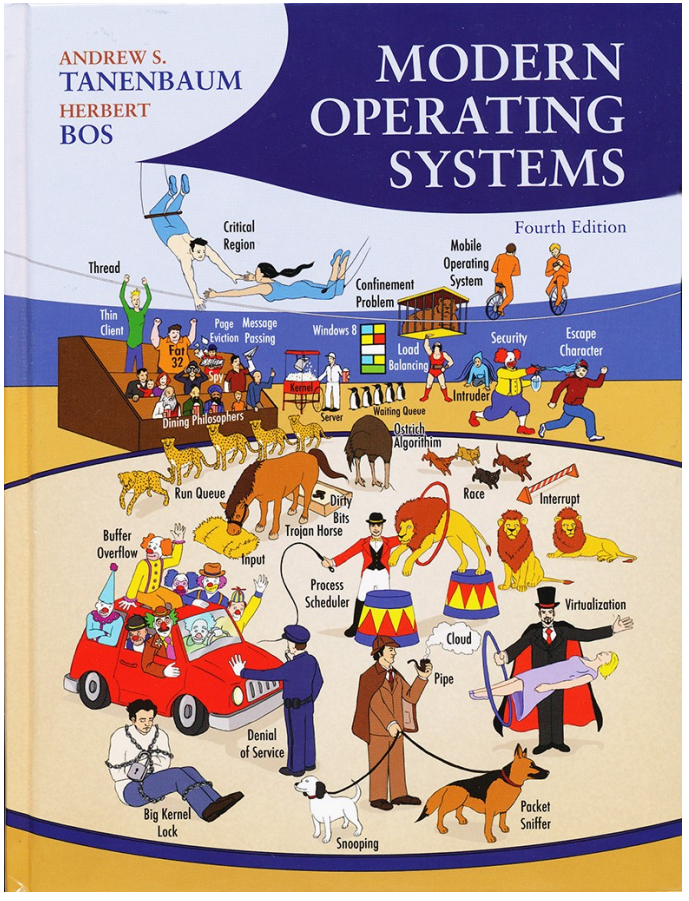
  - A.Silberschatz, P.Galvin, G.Gagne, *Operating Systems Concepts – 10th edition*, JohnWiley & Sons Inc., 2018
  - A.Tanenbaum, *Modern Operating Systems – 4th ed.*, Prentice Hall International, 2015

- Online documentation and resources:

  - http://www.linux.org , etc.

Operating Systems courses





Overview /1

- Basic concepts about OSes
  Hystory of OSes. Types of OSes. Examples.
- OS Architecture
  Components. Services. Design and implementation of Oses. The kernel.
- Process management
  Concepts. Concurrency. Scheduling. Parallel processing.
- Process coordonation
  Critical sections. Syncronization techniques. Inter-process communication. Deadlocks.

Overview /2

- Memory management

  Memory hierarchy. Allocation methods. Segmentation and paging. Virtual memory. Cache memory. Examples.

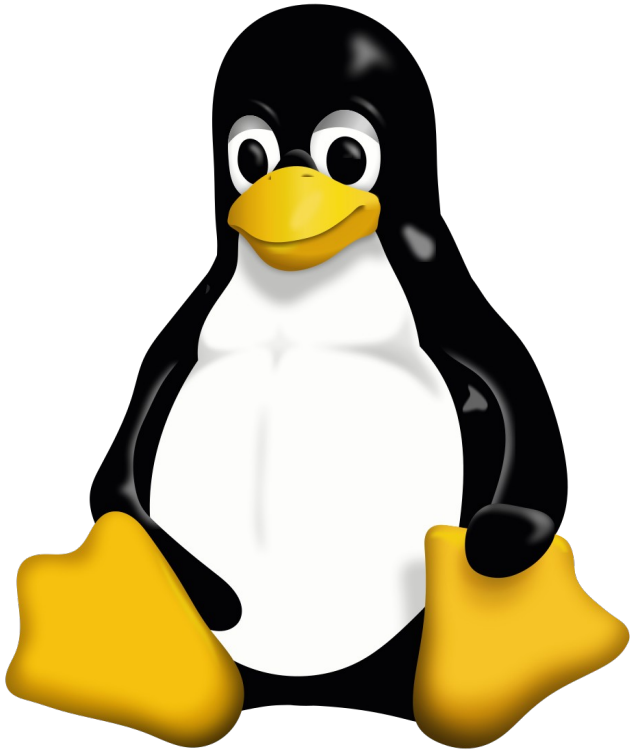
- Storage management

  Mass-Storage devices. Disk management.
  File systems. Design and implementation. Access protection.

- Distributed systems

  Types of distributed OSes. Distributed coordination.
  Distributed file systems.

Operating Systems practical labs



Overview /3

UNIX / Linux OS

- User guide:
  - Basic commands
  - Shell (bash) scripting

- Concurrent Programming in C:
  - Files
  - Processes
  - Inter-process communication (pipes & fifos, signals, I/O redirects)
  - Resource management (terminals, users, etc.)

A parenthesis… /1

Some (new) concepts:

- Sequential programming (classical):

= a program with a single flow of instructions in progress at runtime

vs.

- Parallel, concurrent (sometimes also **distributed**) **programming**:

= a program with multiple streams of instructions in progress at the "same" time during runtime

The N≥2 streams of instructions (called *threads* or *sequencial processes*) run in parallel and, moreover, they compete for the use of the resources provided by the OS.

Sometimes, those N≥2 streams of instructions can be executed in a way distributed in space, i.e. on different computer systems, interconnected by a network.

How can multiple streams of instructions be running at the "same" time?

- by **apparent parallelism**: the *multitasking* technique

vs.

- by **true parallelism**: the *multiprocessing* technique

A parenthesis… /2

Classification by the user interface (abbreviated, UI) of the programs:

- **Programs with Text-based UI** (abbreviated, *TUI*, or *CLI*):

= a program that interacts with the user through a *C*ommand *L*ine *I*nterface

→ the classical **imperative programming** paradigm (or, *program-driven programming*),
with the UI achieved by usual input/output instructions, using only a *keyboard* (+ a screen)

vs.

- **Programs cu Graphical UI** (abbreviated, *GUI*):

= a program that interacts with the user through a graphical interface

→ the **event-driven programming** paradigm,
with the UI achieved by using various libraries/frameworks, with hierarchies made up of hundreds of classes

The graphical interface can be of two kinds:

- *desktop metaphor*: GUI based on WIMP interfaces ("*windows, icons, menus, pointer*"), using a *keyboard* and *mouse*

- *multi-touch metaphor*: GUI based on post-WIMP interfaces, for devices with *touchscreen*

Examination (ro)

I) Evaluare pe parcurs – activitatea la laboratoare (Lab):

1) Prima parte, Utilizarea sistemului Linux:

-- Ex_1: exerciții de laborator săptămânale ; punctaj total (maxim) 5p

-- TP_1 : unul (sau mai multe) teste practice de rezolvat în laborator ; punctaj total (maxim) : 25p ; fiecare dintre ele va fi anunțat cu o săptămână înainte

2) A doua parte, Programare concurentă în C:

-- Ex_2: exerciții de laborator săptămânale ; punctaj total (maxim) 5p

-- TP_2 : unul (sau mai multe) teste practice de rezolvat în laborator ; punctaj total (maxim) : 25p ; fiecare dintre ele va fi anunțat cu o săptămână înainte

Aceste teste practice **nu** se vor da și în restanțe!

(Notă: le veți putea reface în anul următor cu grupa specială de restanțieri)

Punctajul total pentru laborator : $Lab = (Ex_1 + Ex_2) + (TP_1 + TP_2)$

Criteriul de promovare a laboratorului: $Lab \geq 20p$

Examination (ro)

II) Evaluare finală – examen scris (TS) :

- Se va susține la finalul semestrului [sau în restanțe] un test scris cu subiecte din partea de teorie
- Condiții de intrare în examen: promovarea laboratorului și prezența la testul scris
- Punctajul (maxim) TS : 40p
- Criteriul de promovare a examenului: TS ≥ 15p

Bonus (pentru motivarea pregătirii practice):

dacă $TP_1 \geq 23p$ și $TP_2 \geq 22p$, atunci NU mai este obligatoriu să susțineți testul scris, acesta fiind echivalat cu $TS = (TP_1+TP_2)*4/5$

Examination (ro)

III) Nota finală:

- Criteriul de promovare a disciplinei:

să fie promovate laboratorul și examenul scris

- Punctajul final (PF):

PF = TS + Lab
Notă: punctajul final se calculează numai pentru cei care îndeplinesc criteriul de promovare a disciplinei

- Nota finală:

se calculează pe baza PF în conformitate cu criteriile ECTS...

Examination (ro)

III) Nota finală (cont.):

Se calculează pe baza PF în conformitate cu criteriile ECTS...

Acest lucru se referă la faptul că la stabilirea notei finale se ia în considerare distribuția statistică a punctajelor.
Procentele propuse de ECTS ("top 10%, next 25%, next 30%, next 25%, lowest 10%") se referă la cazul translatării notelor dintr-un sistem în altul și se bazează pe faptul că notele urmează distribuția normală (i.e., distribuția gaussiană: "clopotul lui Gauss").

Realitatea din anii anteriori ne arată că distribuția punctajelor obținute de studenți în urma evaluărilor nu urmează întotdeauna distribuția normală. De exemplu, sunt cazuri când 75% (sau chiar mai mult) dintre punctajele celor promovați se află în intervalul [35-60]. Evident că în astfel de cazuri nu se pot aplica procentele mai sus menționate pentru că ar anula practic, de exemplu, diferența dintre cineva cu 80 de puncte și cineva cu 60 de puncte.
Din acest motiv, procentele de mai sus sunt luate doar ca un punct de plecare în stabilirea notelor finale; dacă distribuția punctajelor finale va fi diferită de cea normală, atunci și scala procentajelor trebuie adaptată la noua distribuție.

Feedback

- Discussion (during class or tutoring hours)

- For any comments, remarks, issues relating to OS courses and laboratories

  - email: vidrascu@info.uaic.ro
  - contact the OS team (C.Vidrașcu, R.Benchea, V.Ursu)

Questions?