departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Course Administrivia

Concurrency and Parallelism — 2019-20

Master in Computer Science

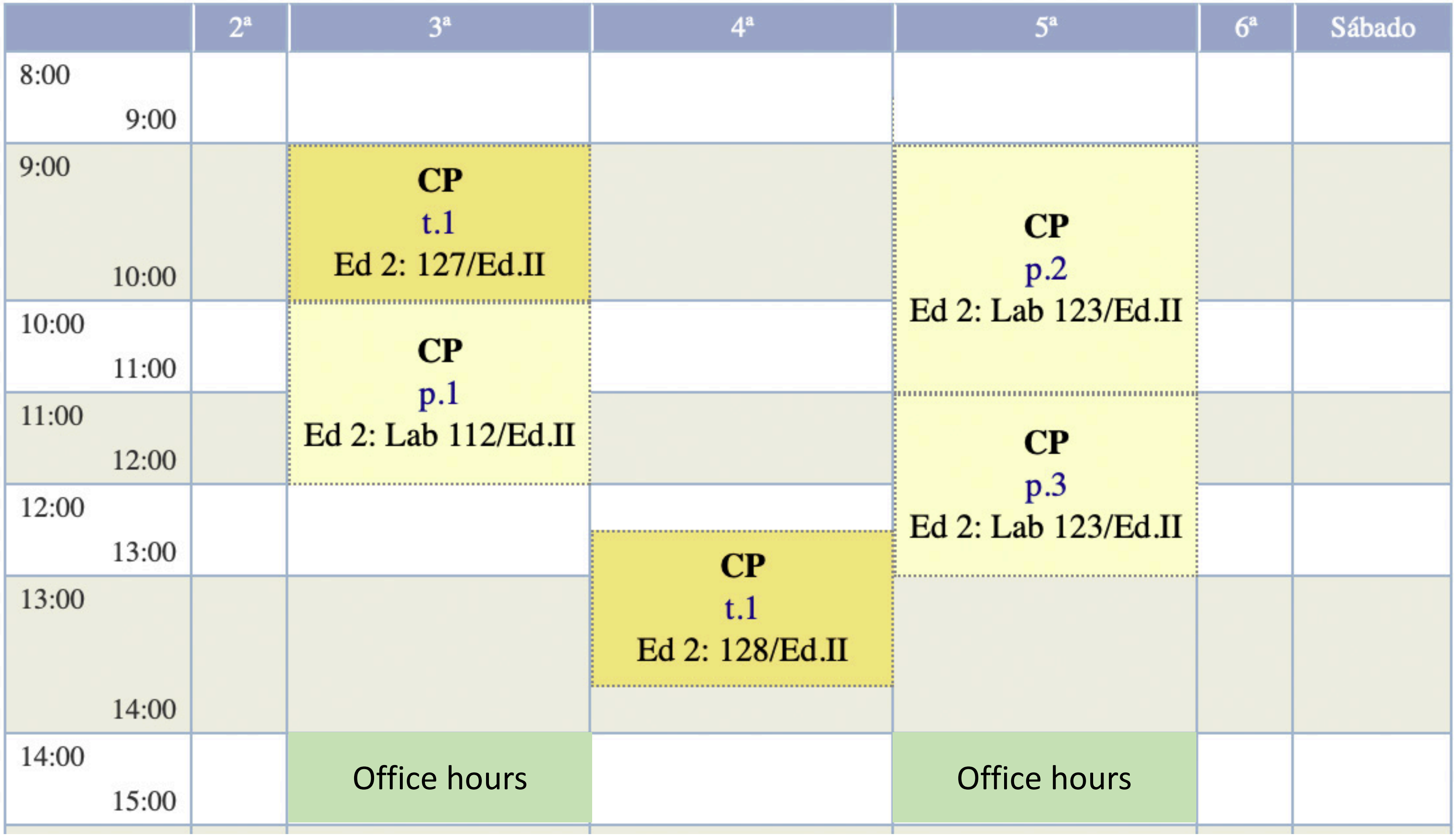
(Mestrado Integrado em Eng. Informática)

Joao Lourenço <joao.lourenco@fct.unl.pt>

## Administrivia — Basic Info

- Lectures
  - João Lourenço <joao.lourenco@fct.unl.pt>
- Labs
  - João Lourenço <joao.lourenco@fct.unl.pt>
- Office location
  - Dep. Informática · Building II · Room P2/9
  - Extension: 10740

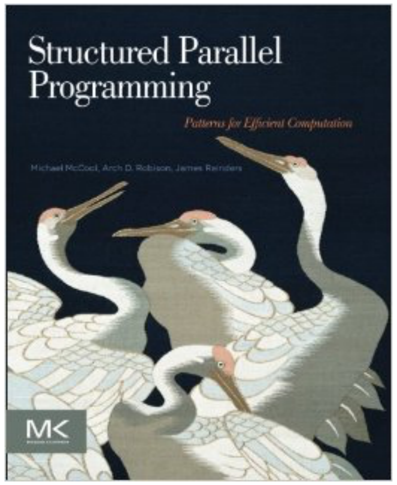
# Administrivia — Schedule



| | 2ª | 3ª | 4ª | 5ª | 6ª | Sábado |
|---|---|---|---|---|---|---|
| 8:00 9:00 | | | | | | |
| 9:00 10:00 | | **CP** t.1 Ed 2: 127/Ed.II | | **CP** p.2 Ed 2: Lab 123/Ed.II | | |
| 10:00 11:00 | | **CP** p.1 Ed 2: Lab 112/Ed.II | | | | |
| 11:00 12:00 | | | | **CP** p.3 Ed 2: Lab 123/Ed.II | | |
| 12:00 13:00 | | | **CP** t.1 Ed 2: 128/Ed.II | | | |
| 13:00 14:00 | | | | | | |
| 14:00 15:00 | | Office hours | | Office hours | | |

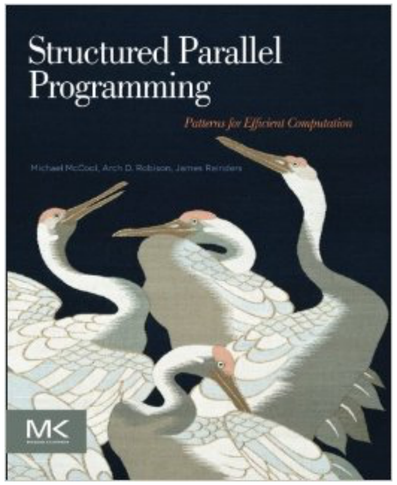
# Administrivia — Main Bib.

- McCool M., Arch M., Reinders J.; **Structured Parallel Programming: Patterns for Efficient Computation**; Morgan Kaufmann (2012); ISBN: 978-0-12-415993-8

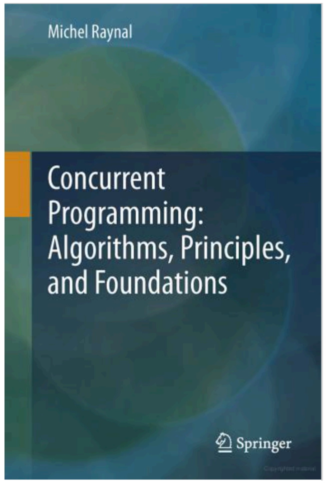

Structured Parallel Programming
Patterns for Efficient Computation
Michael McCool, Arch D. Robison, James Reinders
MK

# Administrivia — Main Bib.

- McCool M., Arch M., Reinders J.; **Structured Parallel Programming: Patterns for Efficient Computation**; Morgan Kaufmann (2012); ISBN: 978-0-12-415993-8

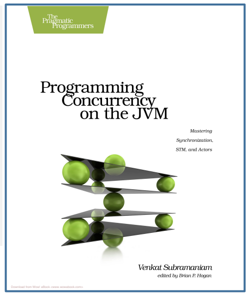

Structured Parallel Programming
Patterns for Efficient Computation
Michael McCool, Arch D. Robison, James Reinders
MK

- Raynal M.; **Concurrent Programming: Algorithms, Principles, and Foundations**; Springer-Verlag Berlin Heidelberg (2013); ISBN: 978-3-642-32026-2


Michel Raynal
Concurrent Programming: Algorithms, Principles, and Foundations
Springer

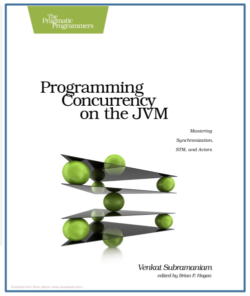
# Administrivia — Additional Bib.

- Suhramaniam V.; **Programming Concurrency on the JVM: Mastering Synchronization, STM, and Actors**; The Pragmatic Bookshelf (2011); ISBN-13: 978-1-934356-76-0

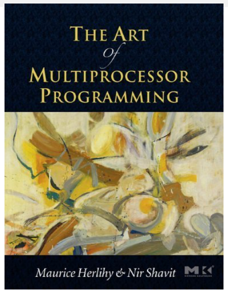

The Pragmatic Programmers
Programming Concurrency on the JVM
Mastering Synchronization, STM, and Actors
Venkat Subramaniam
edited by Brian P. Hogan

# Administrivia — Additional Bib.

- Suhramaniam V.; **Programming Concurrency on the JVM: Mastering Synchronization, STM, and Actors**; The Pragmatic Bookshelf (2011); ISBN-13: 978-1-934356-76-0

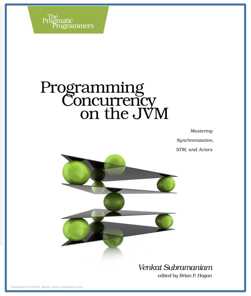


- Herlihy M., Shavit N.; **The Art of Multiprocessor Programming (revised reprint)**; Morgan Kauffman (2012). ISBN: 978-0-12-397337-5

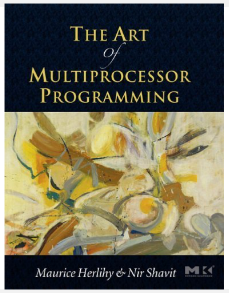


# Administrivia — Additional Bib.

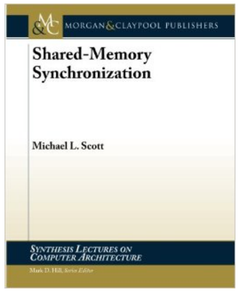
- Suhramaniam V.; **Programming Concurrency on the JVM: Mastering Synchronization, STM, and Actors**; The Pragmatic Bookshelf (2011); ISBN-13: 978-1-934356-76-0


The Pragmatic Programmers
Programming Concurrency on the JVM
Mastering Synchronization, STM, and Actors
Venkat Subramaniam
edited by Brian P. Hogan

- Herlihy M., Shavit N.; **The Art of Multiprocessor Programming (revised reprint)**; Morgan Kauffman (2012). ISBN: 978-0-12-397337-5

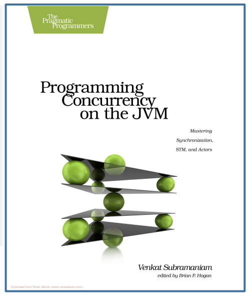

The Art of Multiprocessor Programming
Maurice Herlihy & Nir Shavit

- Michael L. S.; **Shared-Memory Synchronization**; Morgan & Claypool (2013); ISBN: 978-1-608-45956-8

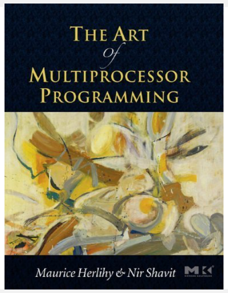

Morgan & Claypool Publishers
Shared-Memory Synchronization
Michael L. Scott
Synthesis Lectures on Computer Architecture
Mark D. Hill, Series Editor

# Administrivia — Additional Bib.

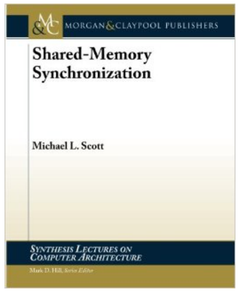
- Suhramaniam V.; **Programming Concurrency on the JVM: Mastering Synchronization, STM, and Actors**; The Pragmatic Bookshelf (2011); ISBN-13: 978-1-934356-76-0


The Pragmatic Programmers
Programming Concurrency on the JVM
Mastering Synchronization, STM, and Actors
Venkat Subramaniam
edited by Brian P. Hogan

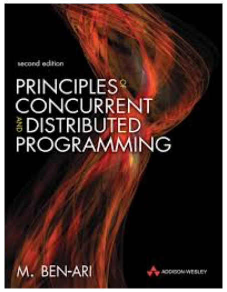
- Herlihy M., Shavit N.; **The Art of Multiprocessor Programming (revised reprint)**; Morgan Kauffman (2012). ISBN: 978-0-12-397337-5


The Art of Multiprocessor Programming
Maurice Herlihy & Nir Shavit

- Michael L. S.; **Shared-Memory Synchronization**; Morgan & Claypool (2013); ISBN: 978-1-608-45956-8

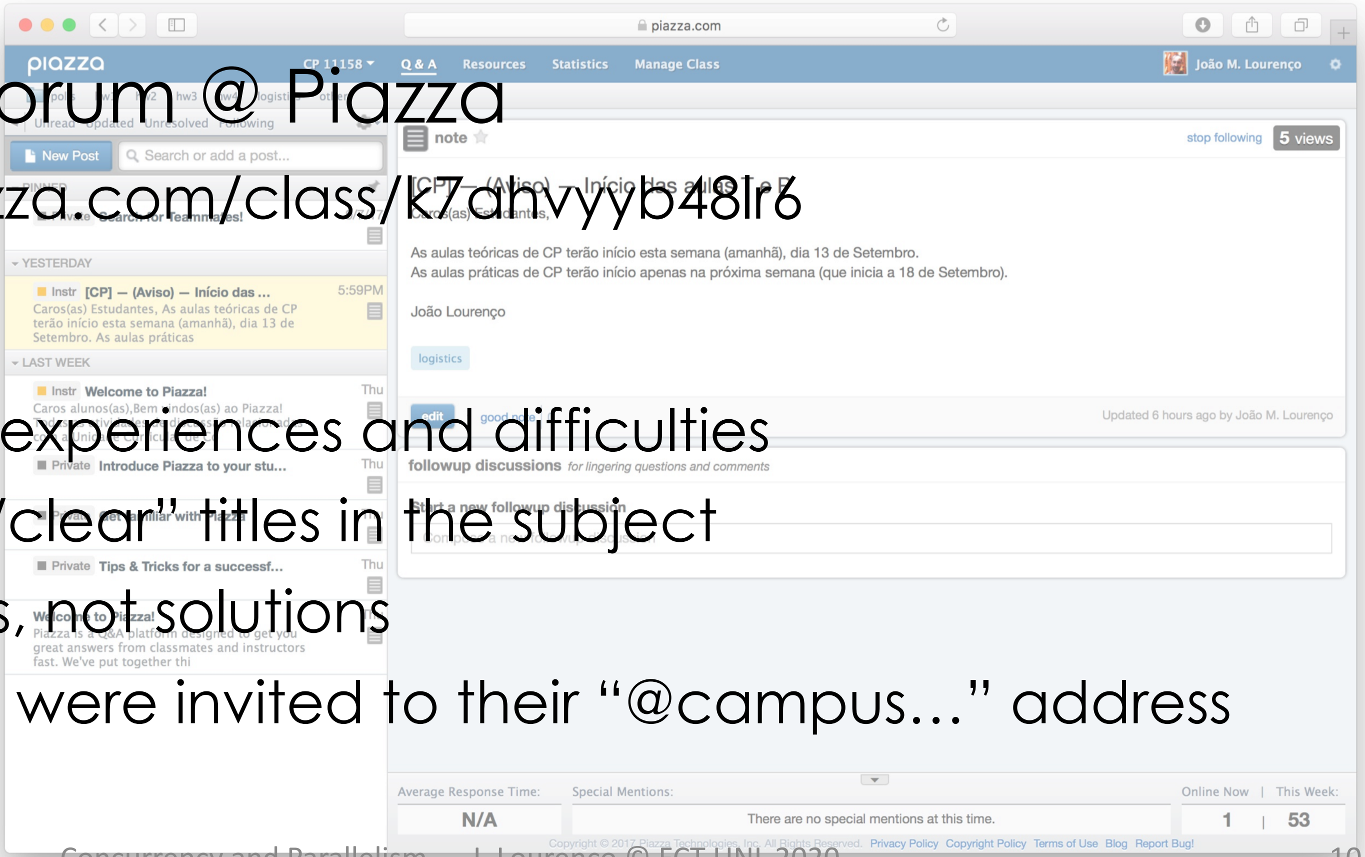

Morgan & Claypool Publishers
Shared-Memory Synchronization
Michael L. Scott
Synthesis Lectures on Computer Architecture
Mark D. Hill, Series Editor

- Ben-Ari M.; **Principles of Concurrent and Distributed Programming, 2/E**; Pearson (2006); ISBN: 978-0-321-31283-9



# Administrivia — Additional Bib.

- Class web page @ CLIP
  - All assignments, handouts, [lecture notes]
- Discussion forum @ Piazza
  - https://piazza.com/class/k7ahvyyb48lr6
- Rules
  - Share your experiences and difficulties
  - Use “smart/clear” titles in the subject
  - Share ideas, not solutions
  - All students were invited to their “@campus…” address



piazza.com

piazza CP 11158 Q & A Resources Statistics Manage Class João M. Lourenço

New Post Search or add a post...

YESTERDAY

Instr [CP] — (Aviso) — Início das ... 5:59PM
Caros(as) Estudantes, As aulas teóricas de CP terão início esta semana (amanhã), dia 13 de Setembro. As aulas práticas

LAST WEEK

Instr Welcome to Piazza! Thu
Caros alunos(as),Bem-vindos(as) ao Piazza!

Private Introduce Piazza to your stu... Thu

Private Tips & Tricks for a successf... Thu

Welcome to Piazza!
Piazza is a Q&A platform designed to get you great answers from classmates and instructors fast. We've put together thi

note stop following 5 views

[CP] — (Aviso) — Início das aulas T e P

Caros(as) Estudantes,

As aulas teóricas de CP terão início esta semana (amanhã), dia 13 de Setembro.
As aulas práticas de CP terão início apenas na próxima semana (que inicia a 18 de Setembro).

João Lourenço

logistics

edit good note Updated 6 hours ago by João M. Lourenço

followup discussions *for lingering questions and comments*

Start a new followup discussion

Average Response Time: N/A
Special Mentions: There are no special mentions at this time.
Online Now | This Week: 1 | 53

Copyright © 2017 Piazza Technologies, Inc. All Rights Reserved. Privacy Policy Copyright Policy Terms of Use Blog Report Bug!

## Administrivia — Course Goals: Knowledge

- To understand the concepts of concurrency and parallelism, and how they can be explored when designing software;
- To identify the models used for problem solving in multiprocessors and highly-parallel systems;
- To know the paradigms used in the development of algorithms for multiprocessors and highly-parallel systems;
- To know the languages, libraries and tools used in the development of concurrent and parallel programs;
- Be familiar with common concurrency problems, and how to mitigate or avoid them.

## Administrivia — Course Goals: Application

- Be able to identify and exploit opportunities for concurrency and parallelization within a software system;
- Be able to partition a problem into multiple tasks to be executed in a parallel system;
- Be able to reason about the behavior of concurrent and parallel programs;
- Be able to build correct and efficient concurrent and parallel algorithms;
- Be able to use the Java/C-like programming languages and parallel libraries to develop parallel software systems;
- Be able to use programming tools in the development of concurrent and parallel applications, including the design, implementation, debugging and deployment stages;
- Be able to predict and measure the performance characteristics of a parallel system.

# Syllabus: Concurrency

1. **Parallel architectures**
   Flynn's taxonomy; performance theory (including Amdahl's and Gustafson's laws).
2. **Parallel programming**
   The spectrum of high-demanding computational problems; regular and irregular problems; strategies for problem decomposition and their mapping to programming patterns; the transactional and map-reduce models.
3. **Concurrency control and synchronization**
   Competition and collaboration; atomicity; linearization; monitors; locks; semaphores; barriers; producer-consumer; multi-reader single-writer locks; futures; concurrency in practice in Java and C.
4. **Safety and liveness**
   Safety vs. liveness; progress; deadlock; deadlock prevention, avoidance, detection, and recovery; livelock; livelock avoidance; priority inversion; priority inheritance. Lock-free algorithms.
5. **The transactional model**
   Composite operations; transactions (serializability), optimistic concurrency control (OCC) and transactional memory.
6. **Concurrency without shared data**
   Active objects; message passing; actors.

# Lab classes

- In the class
  - Design and implement parallel and concurrent programs
- One Homework / Project
  - Addressing parallelism and/or concurrency
- Rules for grouping
  - Group members may be enrolled in different lab classes
  - Groups of 2 students
    - ****All exceptions**** require explicit authorization
    - Non-authorized individual projects ****will not**** be graded

# Administrivia — Evaluation

- [65%] three tests (individual)
  [ average ≥ 9.5 points ]
- [30%] one HW/project (groups of 3 students)
  [ grade ≥ 9.5 points ]
- [5%] participation in class' life cycle
  (includes lectures, labs, piazza, etc)
  (please note that "participation ≠ being there")

The tests will contain questions about the lab exercises and home project

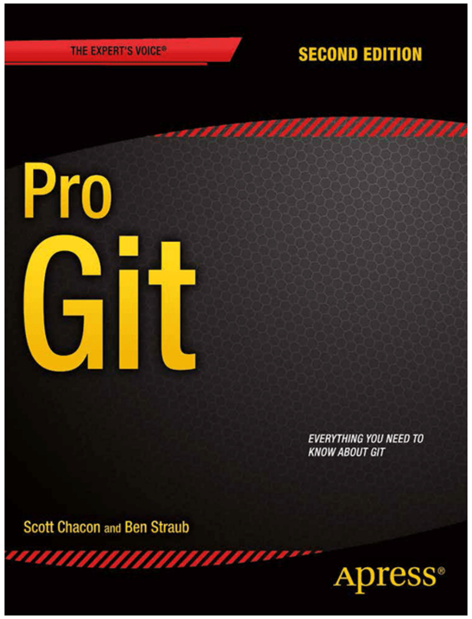
# Administrivia — Frequency

- Frequency:
  - Project ≥ 9.5 points => FREQUENCY ACQUIRED
- Frequency from 2018-19?
  - If used, considered as 30% of the final grade
  - If planning to use, please unroll from the lab class where you are registered
  - Tests will include questions about the lab exercises and this year's project. ***No one is exempted from answering these questions.***
- Frequency from 2017-18 or before?
  - You must acquire frequency again this year!

# Administrivia — Project devlp.

- We will use GIT extensively
- Each group member will commit his/her individual contributions to the group repository
- I expect at least one commit / week / person during the project development
  - Commit logs/messages must clearly state the contributions
- Students will be evaluated by their individual contributions in the GIT repository
  - No meaningful commits/contributions
    => no frequency (fail)
- **Project submission will be a *GIT pull request* (well commented)**
- ***Learn GIT now!!!!***

  *https://git-scm.com/book/en/v2*


THE EXPERT'S VOICE®
SECOND EDITION
Pro
Git
EVERYTHING YOU NEED TO KNOW ABOUT GIT
Scott Chacon and Ben Straub
Apress®

## Administrivia — Project report

- I don't care who does what in the project, as long as everybody does technically relevant / meaningful work for the project
- **Work division must be reported** in the project report
  - Must be supported by the individual commit logs

Any attempt of fraud => all groups members will fail the course immediately

## Administrivia — Project grading

- The project assignment is:
  - Useful to practice what has been read in the text and/or discussed in the class lecture sessions
  - Useful to get some practical exposure to writing code
  - Useful for exploring new material on your own, and working through ideas discussed in class or in the text book
- I expect some level of professionalism in your submissions
  - Professionalism in this case means making sure that every one of your solutions attain a level of neatness, organization, and quality

## Administrivia — Project grading

- Solution must adequately address the problem at hand, i.e.
  - The solution represents a good-faith attempt to actually address the requirements for the assignment.
  - The program complies and executes.
  - The program runs correctly.
- The solution constitutes a high quality product expected of a professional. Specifically:
  - The program is easy to read and to understand, i.e., it is well commented and adheres to the course standards for layout and format. In addition, method and variable names are meaningful, all potentially confusing/complex code is well documented.
  - The general design of the program is clear and reasonable.
  - All procedure and function headers include comments explaining what the method is supposed to do (*not how it does it*) and the purpose of each formal parameter. Be as precise and careful as you can be.

# Administrivia — Project grading

| Max Score | Quality of Project (code and written report) |
|---|---|
| 20 | The project meets or exceeds the expectations outlined before |
| 16 | The project meets most of the expectations outlined before, but fails to meet at least one of them adequately |
| 13 | The project meets some of the expectations outlined before, but fails to meet more than one of them adequately |
| 10 | The project just barely meets the expectations outlined before.  It compiles and runs, but there are serious problems with the code or it does not do everything required by the assignment |
| 0 | The project is unacceptable.  It does not represent a "good faith effort" at a solution, or it does not compile due to syntax errors |

Any project submitted that does not compile and run **WILL RECEIVE AN AUTOMATIC GRADE OF ZERO**.  No exceptions will be made for this rule. To achieve even a single point on a project your code must minimally compile and execute without crashing!

## Administrivia — Project method.

- Feel free to ask questions in/out classes
  - Teacher, colleagues, Piazza
    - *Please make use of Piazza!*
- Fell free to answer questions from colleagues
  - Helping finding a solution ≠ giving the solution for free

- Cite any source that inspired your work
  - As long as you cite what/who you used, it's not cheating
  - Worst case I will deduce some points if it undermines the assignment

# Administrivia — Important Dates

## Year 2019-20 – 2nd semester only

***Março 2020***

| Seg | Ter | Qua | Qui | Sex | Sáb | Dom |
|---|---|---|---|---|---|---|
| | | | | | | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| ***30*** | 31 | | | | | |

***Abril 2020***

| Seg | Ter | Qua | Qui | Sex | Sáb | Dom |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | **10** | 11 | **P** |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | **Expo** | 23 | 24 | **25** | 26 |
| 27 | ***6ª*** | 29 | 30 | | | |

***Maio 2020***

| Seg | Ter | Qua | Qui | Sex | Sáb | Dom |
|---|---|---|---|---|---|---|
| | | | | **1** | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 |

***Junho 2020***

| Seg | Ter | Qua | Qui | Sex | Sáb | Dom |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | **10** | **11** | 12 | **13** | 14 |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 | 30 | | | | | |

***Julho 2020***

| Seg | Ter | Qua | Qui | Sex | Sáb | Dom |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 31 | | |

***Setembro 2020***

| Seg | Ter | Qua | Qui | Sex | Sáb | Dom |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| ***21*** | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | | | | |

🟥 Tests 1, 2 and 3 (to be confirmed)

🟨 Project delivery

## Remember…

- Clip is the official source of information for the course.
- Confirm @Clip all the administrivia related topics.
  - In case of contradiction, is the information in Clip that prevails
- If yours is a special case where the rules are unclear or do not apply, please let me know (*so that we can handle it appropriately*)!

# The END