CSF2600505 Sistem Operasi
CSGE602055 Operating Systems
Week 00: Overview 1

Rahmat M. Samik-Ibrahim (ed.)

University of Indonesia

https://os.vlsm.org/
Always check for the latest revision!

REV213 23-Jan-2020

Operating Systems 2020-1

A [08-10, Rm 3113, Mo/We] — B/M [10-12, Rm 3113, Mo/We] — C [13-15, Rm 3113, Mo/We]

D [10-12, Rm 2307(Mo), Rm 3113(We)] — E [08-10, Rm 2307(Mo), Rm 3113(We)]

Week	Schedule	Topic	OSC10
Week 00	27 Jan - 02 Feb 2020	Overview 1, Virtualization & Scripting	Ch. 1, 2, 18.
Week 01	03 Feb - 09 Feb 2020	Overview 2, Virtualization & Scripting	Ch. 1, 2, 18.
Week 02	10 Feb - 16 Feb 2020	Security, Protection, Privacy, & C-language	Ch. 16, 17
Week 03	17 Feb - 23 Feb 2020	File System & FUSE	Ch. 13, 14, 15
Week 04	24 Feb - 01 Mar 2020	Addressing, Shared Lib, & Pointer	Ch. 9
Week 05	02 Mar - 08 Mar 2020	Virtual Memory	Ch. 10
Reserved	09 Mar - 13 Mar 2020	Q & E	
MidTerm	14-21 Mar 2020 (TBA)	MidTerm (UTS)	Subject to change.
Week 06	23 Mar - 31 Mar 2020	Concurrency: Processes & Threads	Ch. 3, 4
Week 07	01 Apr - 07 Apr 2020	Synchronization & Deadlock	Ch. 6, 7, 8
Week 08	08 Apr - 14 Apr 2020	Scheduling + W06/W07	Ch. 5
Week 09a	15 Apr - 19 Apr 2020	Storage, Firmware, Bootldr, & Systemd	Ch. 11
OCL	20 Apr - 26 Apr 2020	OnLine & CoLearnIng	
Week 09b	27 Apr - 28 Apr 2020	Storage, Firmware, Bootldr, & Systemd	Ch. 11
Week 10	29 Apr - 05 May 2020	I/O & Programming	Ch. 12
Reserved	06 May - 10 May 2020	Q & A	
Final	11-18 May 2020 (TBA)	Final (UAS)	This schedule is subject to change.
Extra	25 Jun 2020	Extra assignment confirmation	

STARTING POINT — https://os.vlsm.org/

- **Text Book** — Any recent/decent OS book. Eg. (**OSC10**) Silberschatz et. al.: **Operating System Concepts**, 10^{th} Edition, 2018. See also http://codex.cs.yale.edu/avi/os-book/OS10/.
- **Resources**
  - **All In One** — BADAK.cs.ui.ac.id:///extra/ (**FASILKOM only!**).
  - **Download Slides and Demos from GitHub.com** https://github.com/UI-FASILKOM-OS/SistemOperasi/
  - **Problems** — https://rms46.vlsm.org/2/: 195.pdf (W00), 196.pdf (W01), 197.pdf (W02), 198.pdf (W03), 199.pdf (W04), 200.pdf (W05), 201.pdf (W06), 202.pdf (W07), 203.pdf (W08), 204.pdf (W09), 205.pdf (W10).

Agenda

1. Start
2. Schedule
3. Agenda
4. How to contact the Lecturer
5. Highlights
6. Week 00
7. Assessment
8. Week 00: Review
9. Assignment (W00) #1: GitHub.com Repository
10. Assignment (W00) #2: GitHub.com Log
11. Assignment (W00) #3: Generate your QR Code
12. Assignment (W00) #4: MEMO Week00
13. Assignment (W00) #5: Try Demo Week00
14. TIPS

Agenda (2)

How to contact the Lecturer[2]

- For Q & A, use WhatsApp Group **OperatingSystems** (info +62-881-456-**XXXX**)
  Email (Subject:**[HELP]**) operatingsystems@vlsm.org
  State your "Name", "ID", and "OS class".

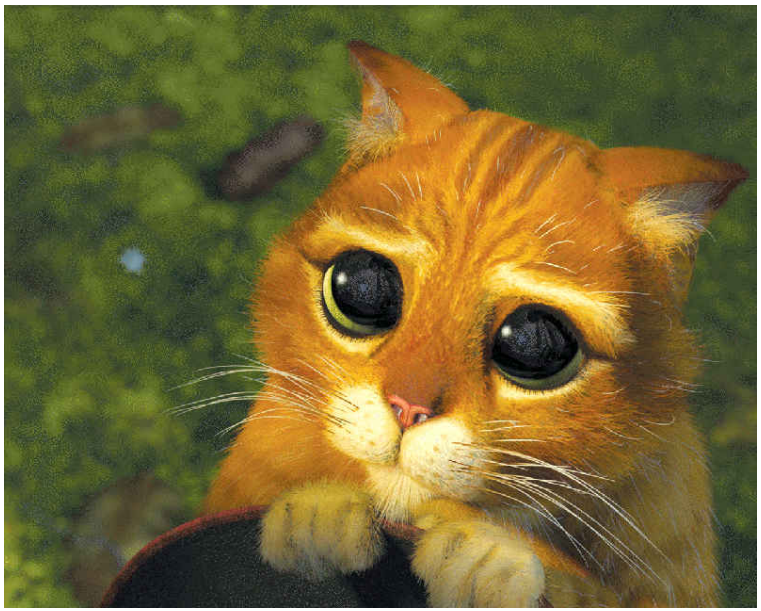


Figure: Never ever whine and pretend like this[1]!

[1] "Puss in Boot" is a DreamWorks/Paramount Picture character.
[2] FYI: King Goerge II founded the University of Goettingen in 1734.

Highlights

Coverage

This is an introduction to a modern operating systems course. It will cover general overview, computer architecture review, operating system overview, GNU/Linux CLI, scripting, C language overview, protection, security, privacy, systemd, I/O, addressing and pointers, memory management, processes and threads, virtual memory, synchronization, mutual exclusion, deadlock, CPU scheduling algorithms, file systems, and I/O programming.

Student-Centered

This course is student-centered where responsibility is in the hands of the students. Students are expected to be prepared for the class meeting.

GNU/Linux

Students will have a thorough understanding of how GNU/Linux provides services by using a Command Line Interface.

Week 00 Overview I: Topics[1]

- Role and purpose of the operating system
- Functionality of a typical operating system
- Mechanisms to support client-server models, hand-held devices
- Design issues (efficiency, robustness, flexibility, portability, security, compatibility)
- Influences of security, networking, multimedia, windowing systems
- Structuring methods (monolithic, layered, modular, micro-kernel models)
- Abstractions, processes, and resources
- Concepts of application program interfaces (APIs)
- The evolution of hardware/software techniques and application needs
- Device organization
- Interrupts: methods and implementations
- Concept of user/system state and protection, transition to kernel mode

[1] Source: ACM IEEE CS Curricula 2013

Week 00 Overview I: Learning Outcomes (1)[1]

- Explain the objectives and functions of modern operating systems [Familiarity]
- Analyze the tradeoffs inherent in operating system design [Usage]
- Describe the functions of a contemporary operating system with respect to convenience, efficiency, and the ability to evolve. [Familiarity]
- Discuss networked, client-server, distributed operating systems and how they differ from single user operating systems. [Familiarity]
- Identify potential threats to operating systems and the security features design to guard against them. [Familiarity]
- Explain the concept of a logical layer. [Familiarity]

[1] Source: ACM IEEE CS Curricula 2013

Week 00 Overview I: Learning Outcomes (2)[1]

- Explain the benefits of building abstract layers in hierarchical fashion. [Familiarity]
- Describe the value of APIs and middleware. [Assessment]
- Describe how computing resources are used by application software and managed by system software. [Familiarity]
- Contrast kernel and user mode in an operating system. [Usage]
- Discuss the advantages and disadvantages of using interrupt processing. [Familiarity]
- Explain the use of a device list and driver I/O queue. [Familiarity]

[1] Source: ACM IEEE CS Curricula 2013

Assessment part 1

85 - ... = A	80 - 85 = A-	75 - 80 = B+	70 - 75 = B
65 - 70 = B-	60 - 65 = C+	55 - 60 = C	50 - 55 = D or C[1]
40 - 50 = D	30 - 40 = E	20 - 30 = E	00 - 20 = E

- **4 SKS** (Units) = 12 hours per week!
  - Ah Beng said: Work hard!
- **No Lab — No Task — No Pop Quiz – No Teaching Assistant**[1].
  - No secret hand-shake!
  - But, it may vary from class to class.
- **Active Preparation / Participation / Q&A Only**.
  - Pre-Midterm (UTS): 6 weeks @ 3 points (=18%).
  - Post-Midterm: 5 weeks @ 3 points (=15%).
  - Points for answering questions, trying demos, and writings memos.
  - Deductions for **NOT** answering questions: individually or collectively.

[1] Terms and Conditions apply. Void where prohibited by law.

Assessment part 2

- **How to get points?**
  - Answer questions, especially not in the middle of a lecture!
  - Try Demos.
  - Class attendance.
- **MidTerm+Final:** $(6 + 5)$ set problems @ 6 points ($= 36\% + 30\%$).
- **Extra Rounding:** 1 point[1]
  - Only if your grade is more than 59.0 and **ONE** more point.
- **C-2C:** upto 5 points[1].
  - Only if your grade is between 50.00 and 55.00 and you have a "good" track record.
- Check your points regularly at `https://academic.ui.ac.id/` and **DO NOT COMPLAIN** weeks after! See also, `https://os.vlsm.org`.

[1] Terms and Conditions apply. Void where prohibited by law.

AIN'T DIFFICULT, lah!

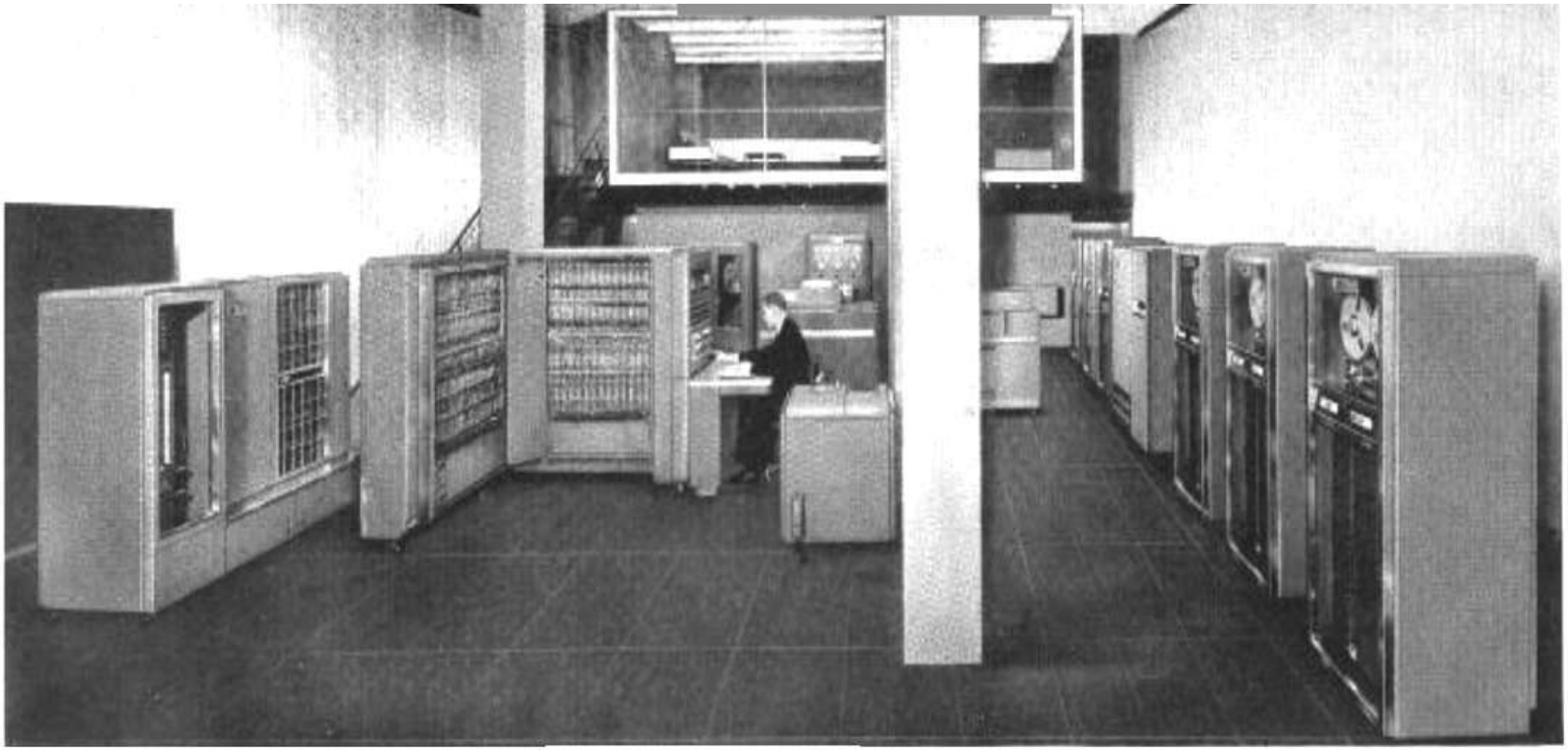
Figure: Even this Goat will get "C" at the end of the semester!

Prelude: Daisy Bell – Bicycle Built for Two

Daisy, Daisy,
Give me your answer, do!
I'm half crazy,
All for the love of you!
It won't be a stylish marriage,
I can't afford a carriage,
But you'll look sweet on the seat
Of a bicycle built for two!

See also https://youtu.be/TXK_cE9AqAI.

IBM 704 at Los Alamos National Laboratory in the 1950s



IBM 704 ELECTONIC DATA-PROCESSING MACHINES

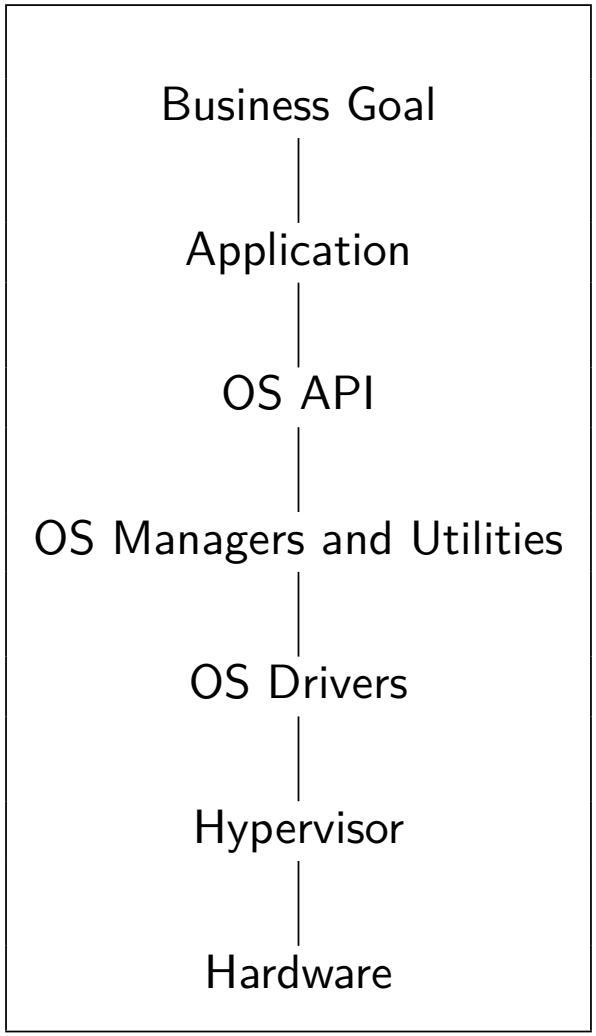
Estimate price (2020 value): USD 8,000,000.

QS855, 256GB, 12 GB, 48+12 MP, 6.4”, 4000 mAh

Estimate price (2020 value): Rp. 8,000,000.

Week 00: Review

- What is an Operating System?
- Why taking an Operating System class?



Computer Organization Review

- You should understand:
  - von Neumann Model.
  - Buses, Bridges, Transfer Rate, Clock.
  - Memory: DDR, DDR-2, DDR-3 ...
  - Cache, Buffer, Spool, & Pipelining.
  - Direct Memory Access (DMA).
  - Port & Memory Mapped I/O.
  - CPU: (privilege/kernel/supervisor mode) vs. (user mode).
  - Physical (Hardware) Limitation.
  - Priority: Read vs Write.
  - Interrupts: Polling & Vectored.
  - Multiprocessors: Symmetric vs. Asymmetric.
  - Multicore & Multithreading.
  - Clustered Systems.
  - Numbers: base 2, base 8, base 10, base 16.
    - Base 2: 110010101010_2
    - Base 8: $01234567_8 = 000\ 001\ 010\ 011\ 100\ 101\ 110\ 111_2$
    - Base 10: 012 345 679
    - Base 16: $9AB\ CDEF_{16} = 1001\ 1010\ 1011\ \ 1100\ 1101\ 1110\ 1111_2$

Block Diagram

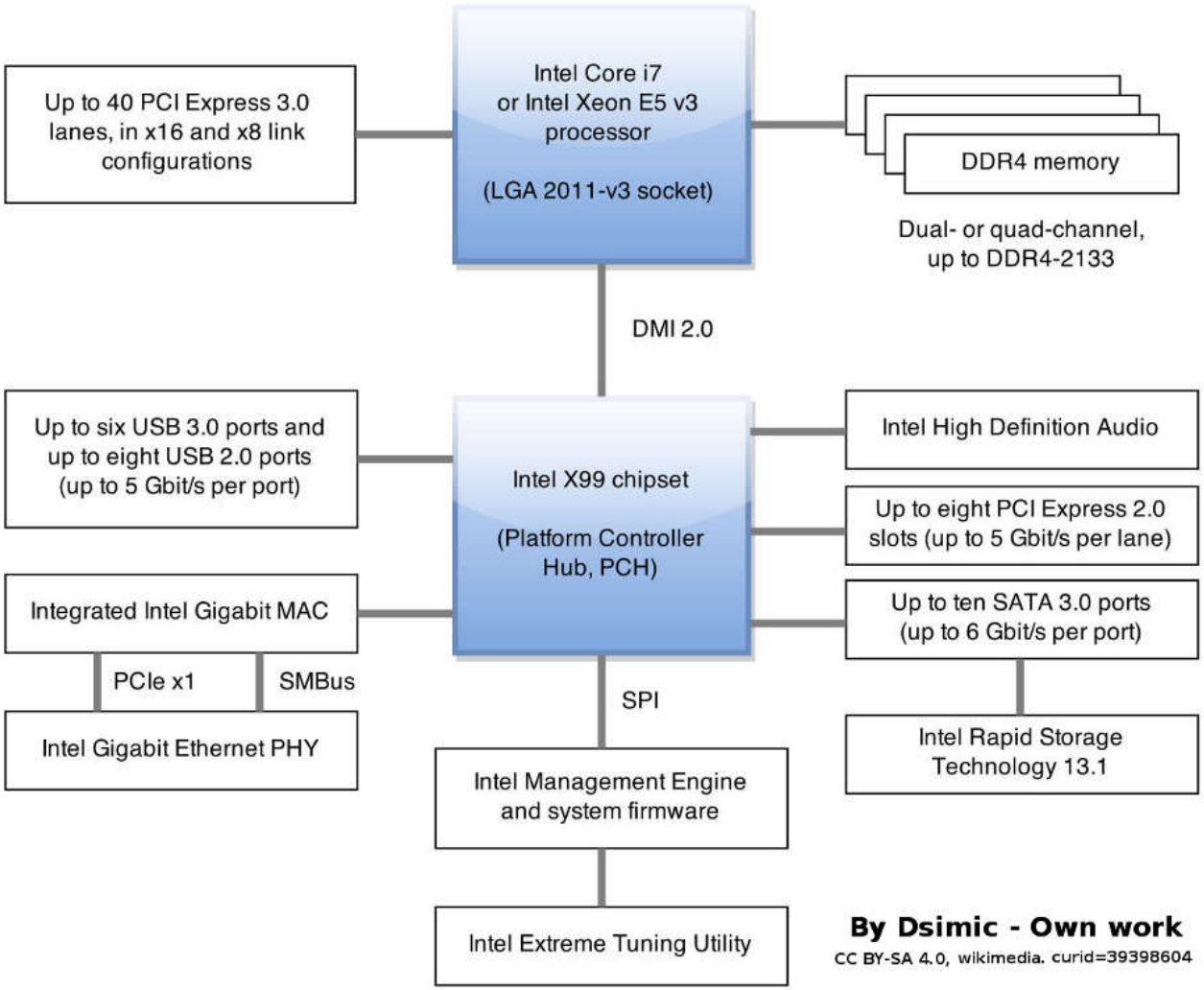


Figure: Block Diagram

APIC (Advanced Programmable Interrupt Controller)

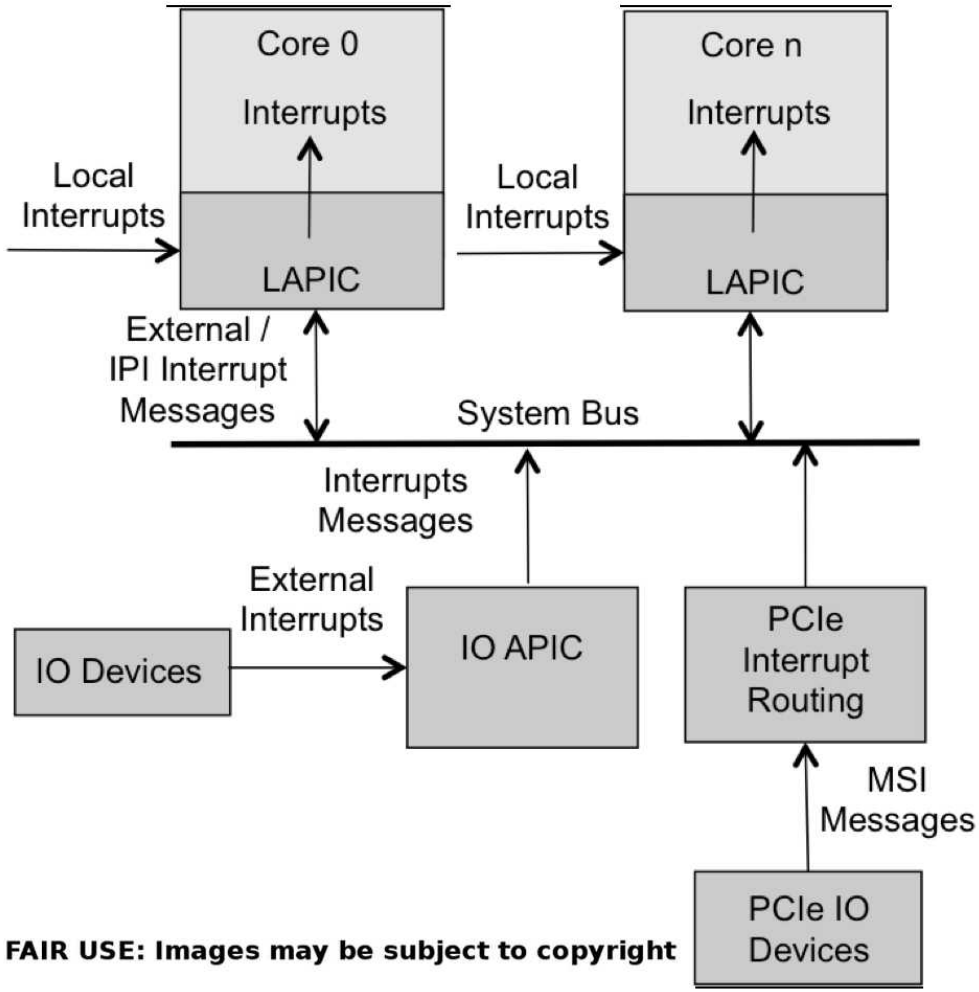


Figure: APIC (Advanced Programmable Interrupt Controller)

Interupt Handling

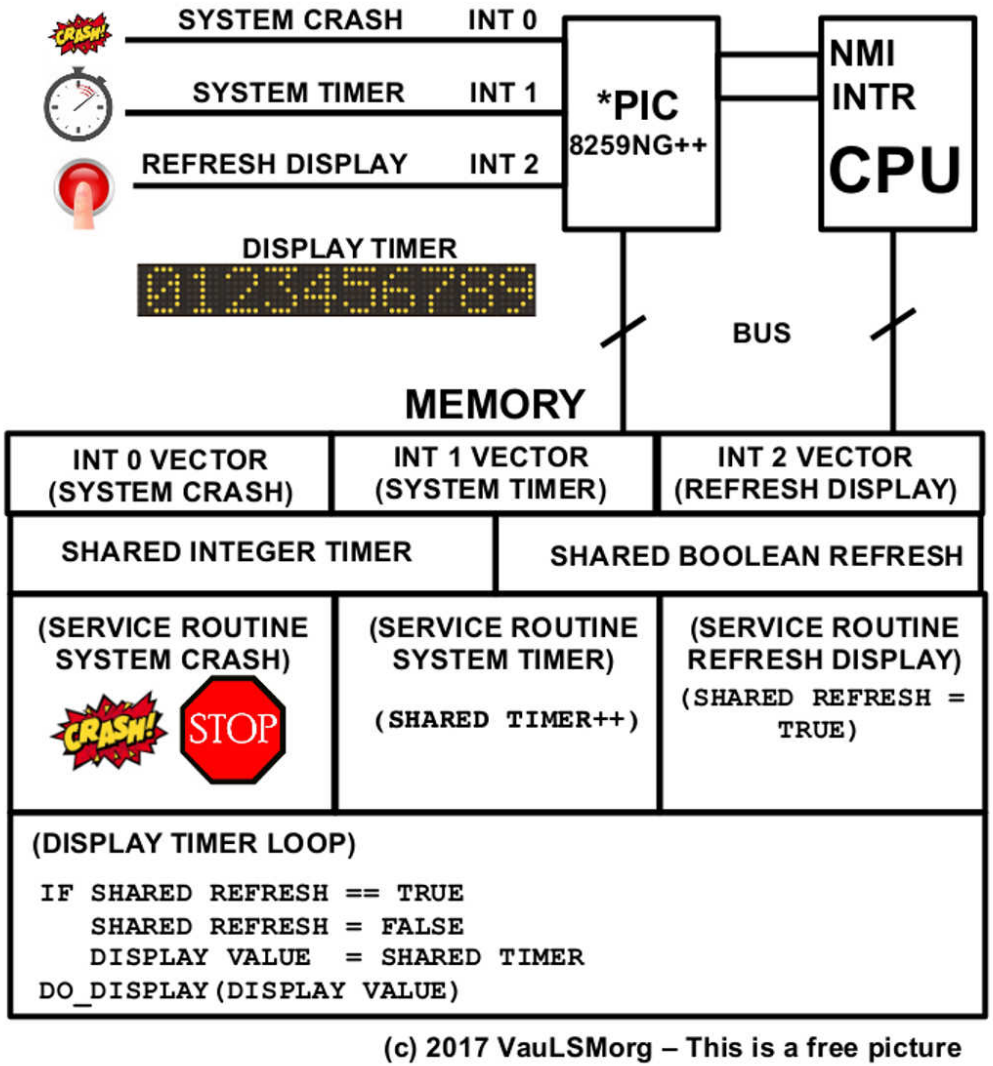


Figure: Interupt Handling with PIC (Programmable Interrupt Controller)

Managers Set

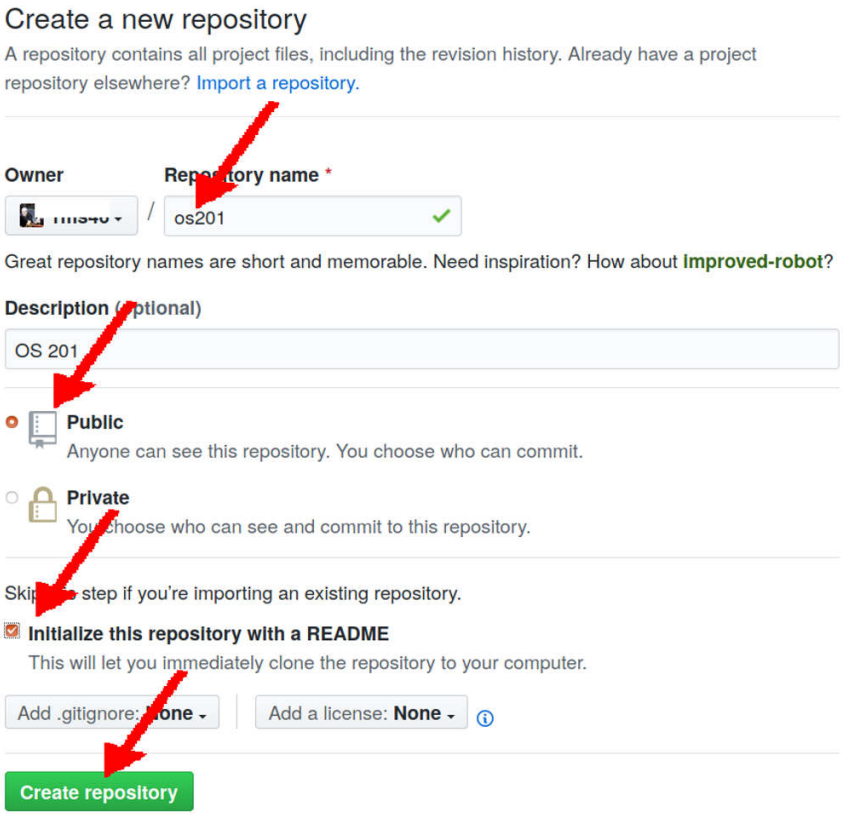
- Process:
  - Creating/Deleting; Suspending/Resuming; Synchronization; Communication; Schedulling
- Memory:
  - Tracking; Move In/Move Out; Allocating/Deallocating.
- Storage/File System:
  - Create/Delete; Open/Close; Read/Write.
- Mass Storage:
  - Schedulling; Allocating; Free Space.
- I/O:
  - Buffering; Caching; Spooling.
  - Interfacing (driving).
- Protecting & Security:
  - Protecting.
  - Security.

Make sure, to understand:

- Scripting: bash, regex, sed, awk?
- Security and Protection?
- File System?
- Data Structure in a (logical) Memory?
- Virtual Memory
- Concurrency
- Synchronization
- Mass Storage
- UEFI, GRUB, and systemd
- I/O
- I/O Programming

Assignment (W00) #1: GitHub.com Repository

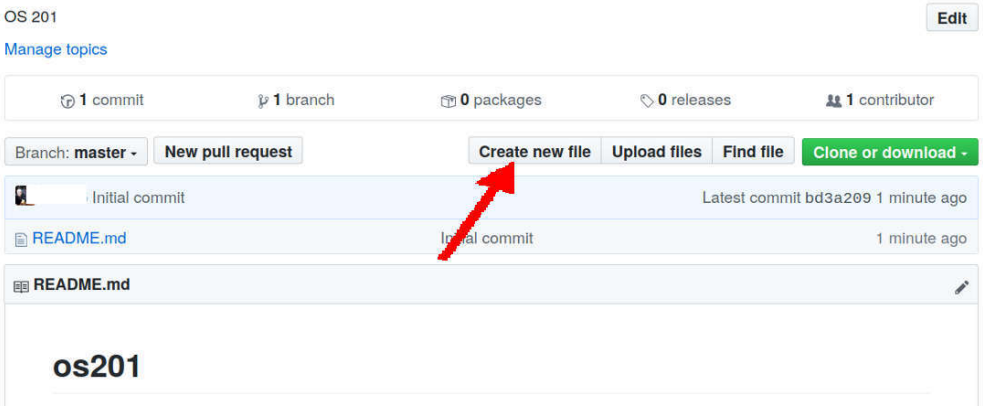
- Visit `github.com`
  - (IF NO ACCOUNT) **SIGN UP** (`https://github.com/join`).
  - (ELSE) **SIGN IN** (`https://github.com/login`).
  - Create a new repository



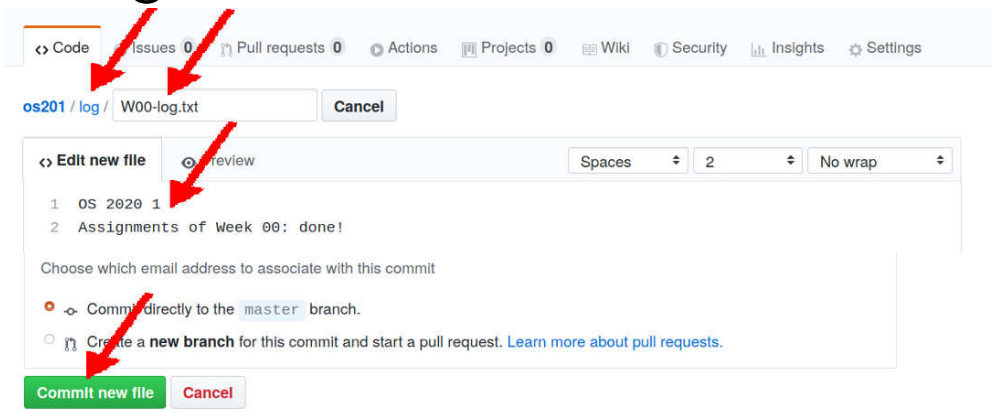
    - **Repository name**: "os201" (all small).
    - **Public**: Anyone can see this repository.

Assignment (W00) #2: GitHub.com Log

- Create a new file inside a new folder



- Folder: "**log**", File: "**W00-log.txt**"
- Write: "`OS 2020 1`
  `Assignments of Week 00: done!`", or something like that.



- **CLICK**: "Commit changes".

Assignment (W00) #3: Generate your QR Code

- Encode your **QRC** with size upto 5cm x 5cm (ca. 400x400 pixels)[1]:

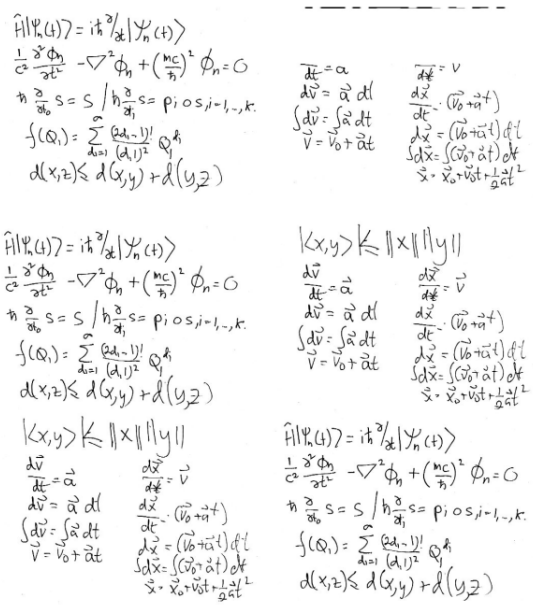
  "OS192 CLASS ID GITHUB-ACCOUNT Your-Full-Name"

  - What year and term? Eg. $2019 - 2 \rightarrow$ "`OS192`"
  - What is your OS class? Regular (A, B, C, D, E)? Or, Extension (X)? Or, International (I)? Or Matrix (M)? Eg. "`X`".
  - What is your Student ID (NPM)? Eg. "`1253755125`".
  - What is your GitHub Account? Eg. "`demo`".
  - What is your Full Name (at SIAK)? Eg. "`Demo Suremo`".
- E.g.: **OS192 X 1253755125 demo Demo Suremo**

Assignment (W00) #4: MEMO Week00

- Write your Memo (with QRC) **every week**.
- Good start: check the previous problems collection.

[OS192][WEEK: 00 01 02 03 04 05 06 07 08 09 10]
[CLASS: A B C D E I M X][ID: 1253755125][Name: Demo Suremo][Rev: 09]



Assignment (W00) #5: Try Demo Week00

- Login to `badak.cs.ui.ac.id` (via `kawung.cs.ui.ac.id`)

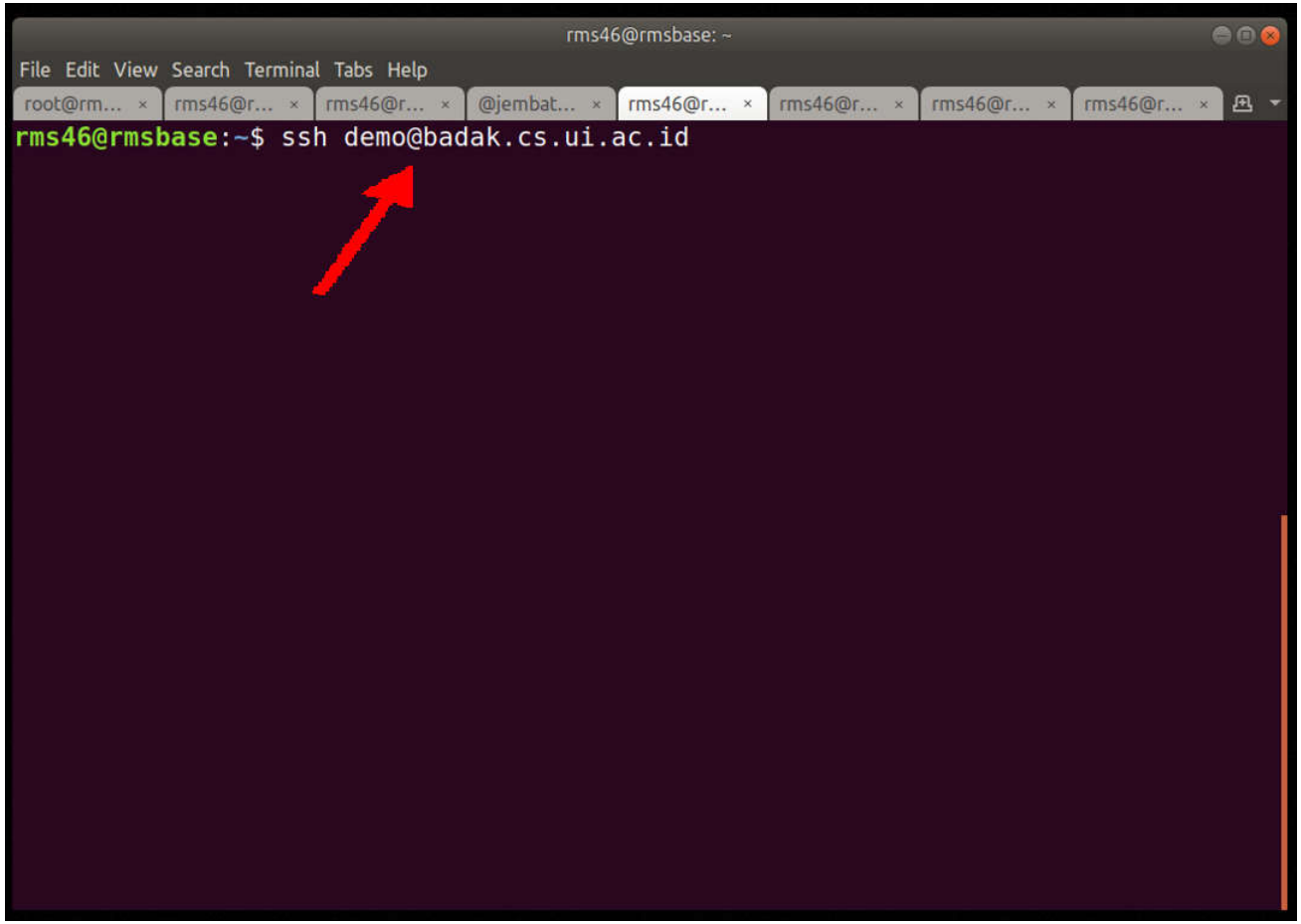


Figure: Login to badak.cs.ui.ac.id using "ssh" Ubuntu GNU/Linux

SSH to "badak.cs.ui.ac.id" — 2

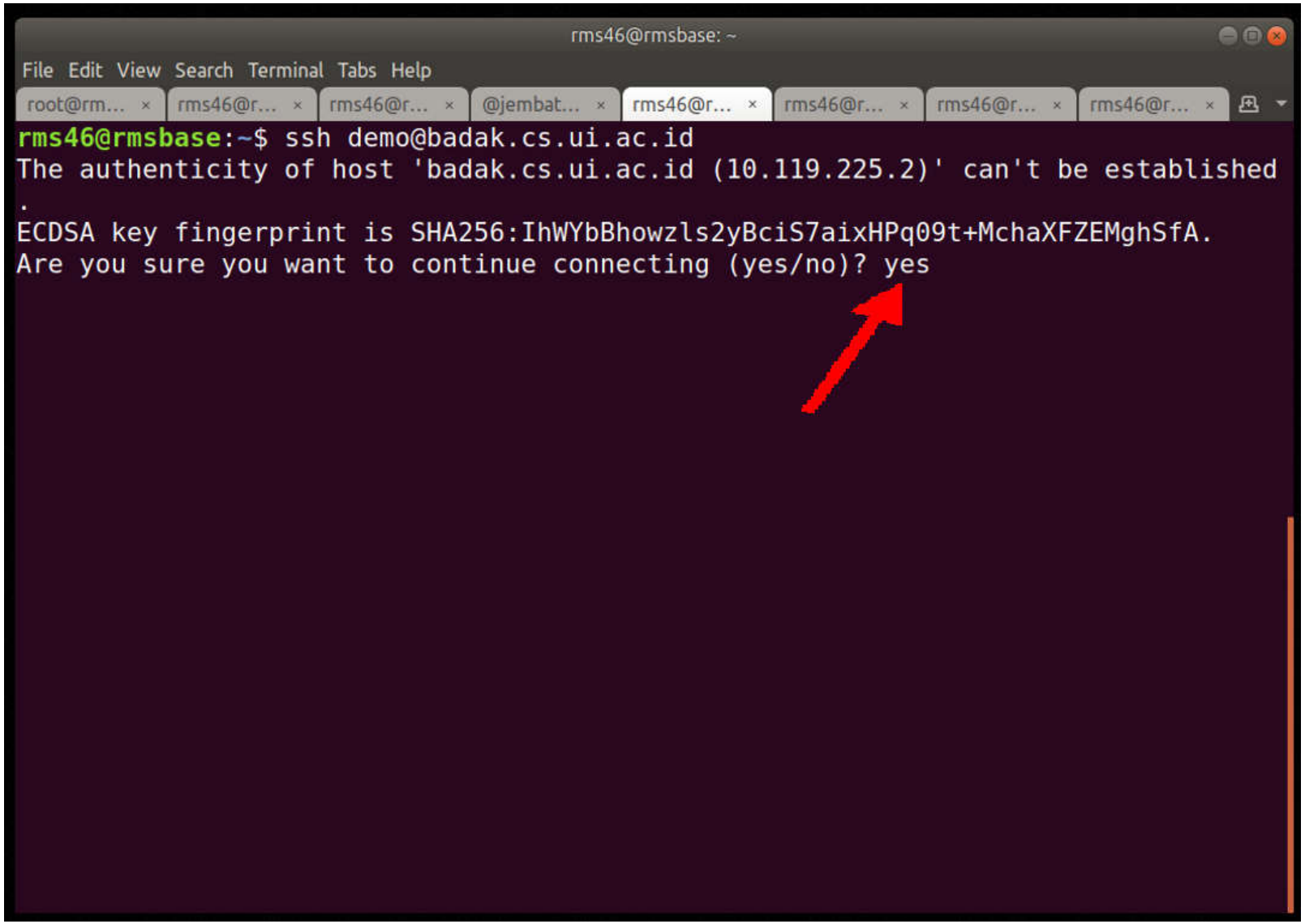


Figure: SSH to "badak.cs.ui.ac.id" — **(For the first time only!)**

SSH to "badak.cs.ui.ac.id" — 3

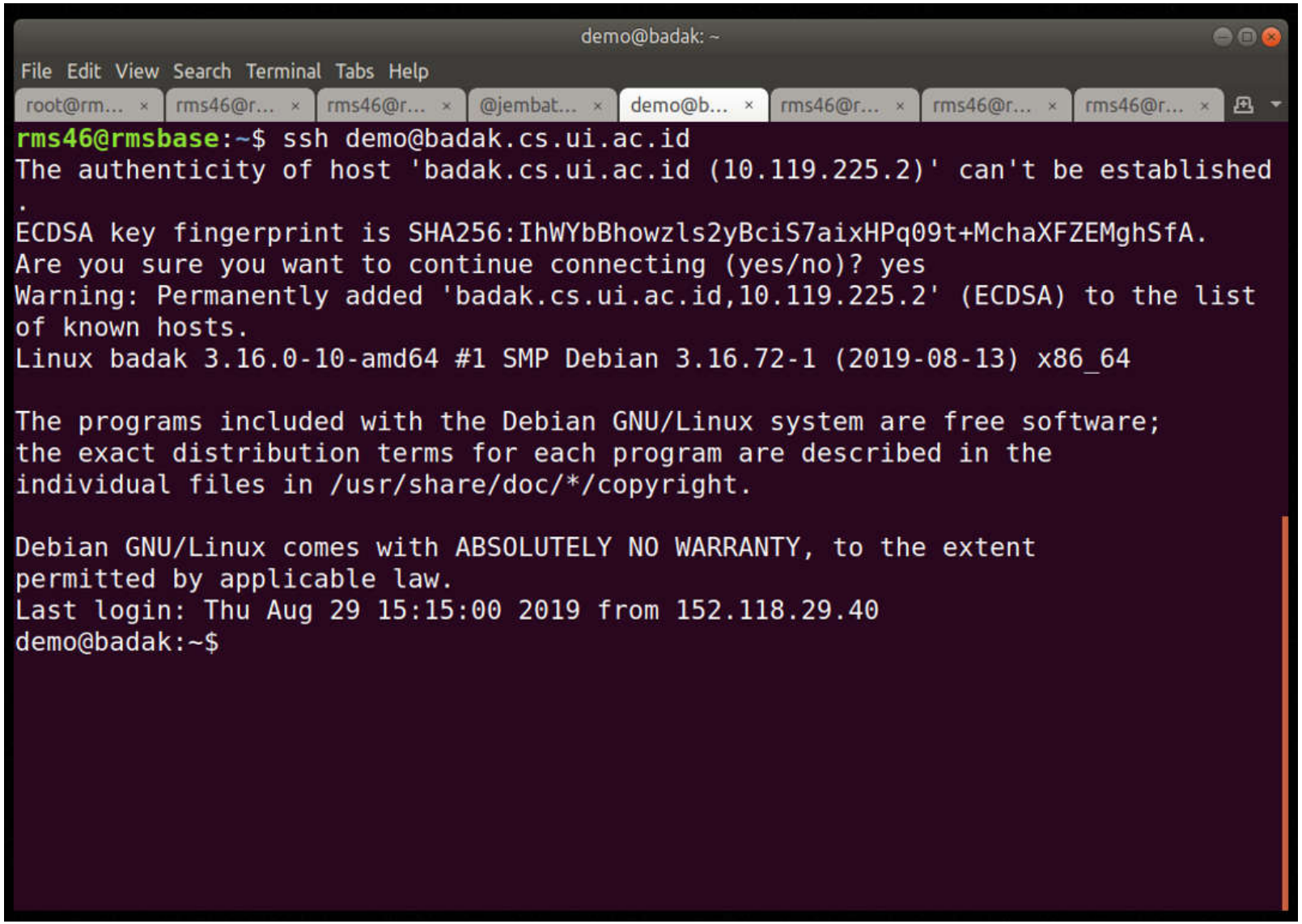


Figure: SSH to "badak.cs.ui.ac.id"

SSH to "badak.cs.ui.ac.id" — 4

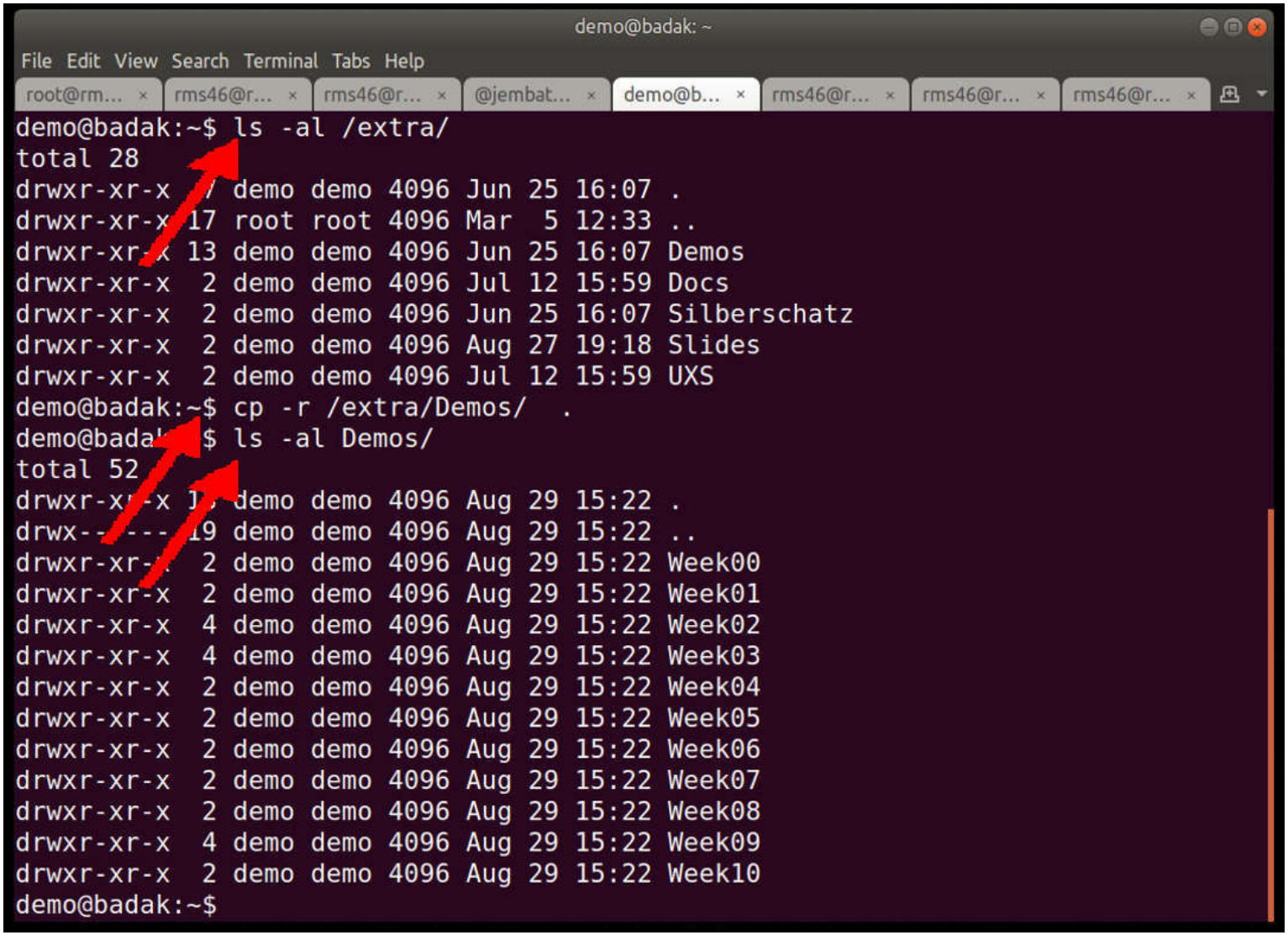


Figure: SSH to "badak.cs.ui.ac.id"

SSH using PowerShell (Windows10) — 1

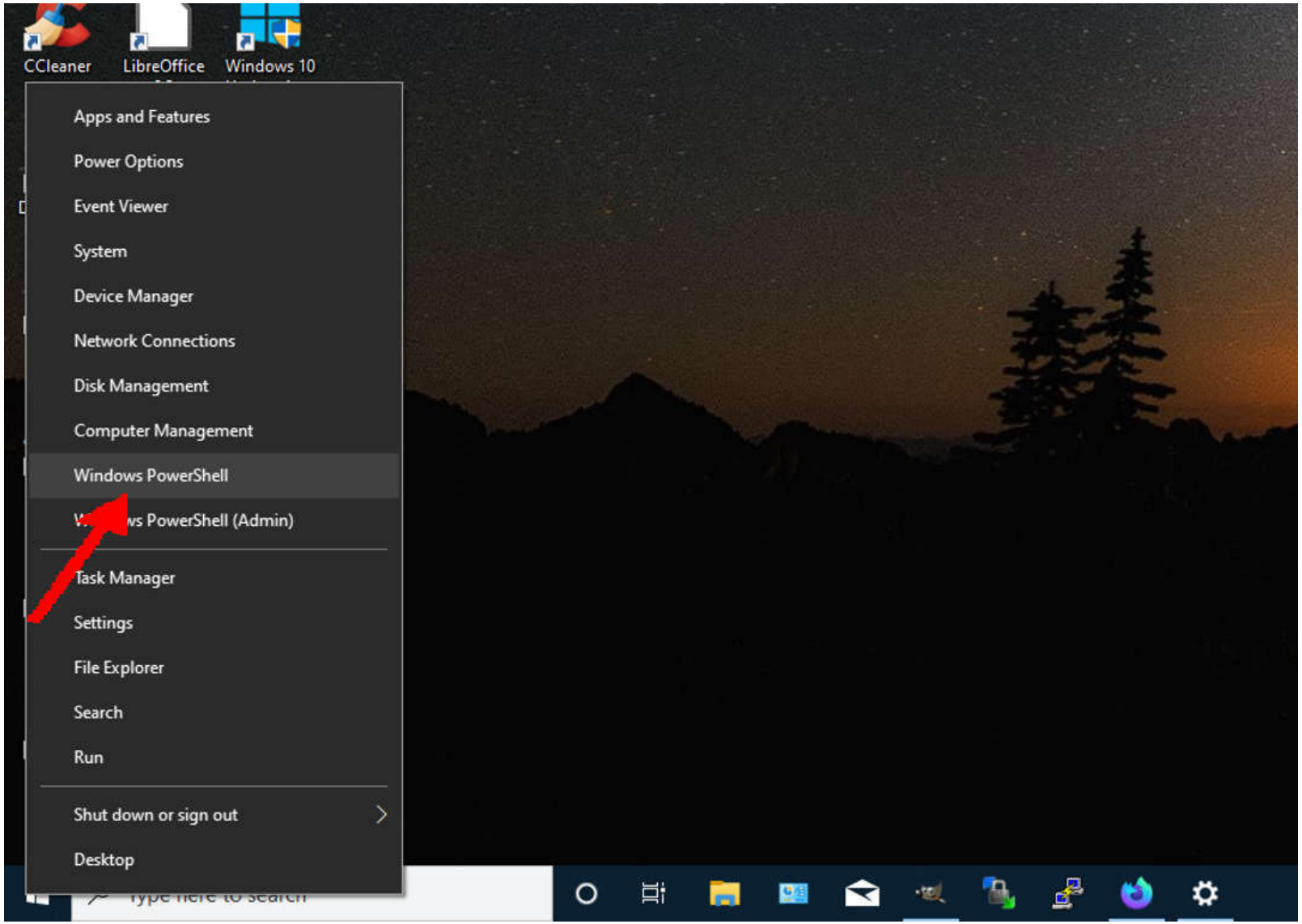


Figure: SSH using PowerShell (Windows 10)

SSH using PowerShell (Windows10) — 2

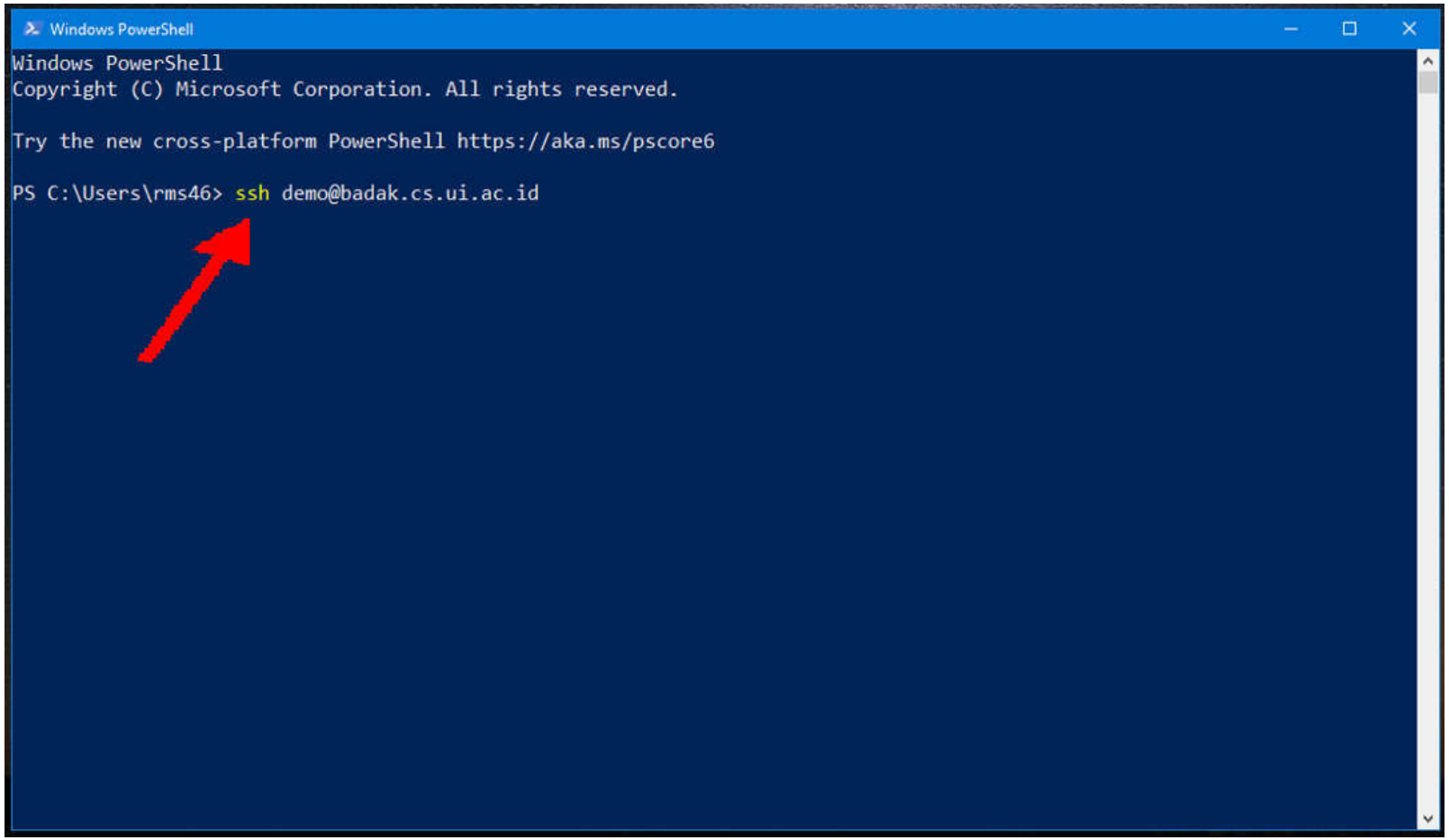


Figure: SSH using PowerShell (Windows 10)

SSH using PowerShell (Windows10) — 3

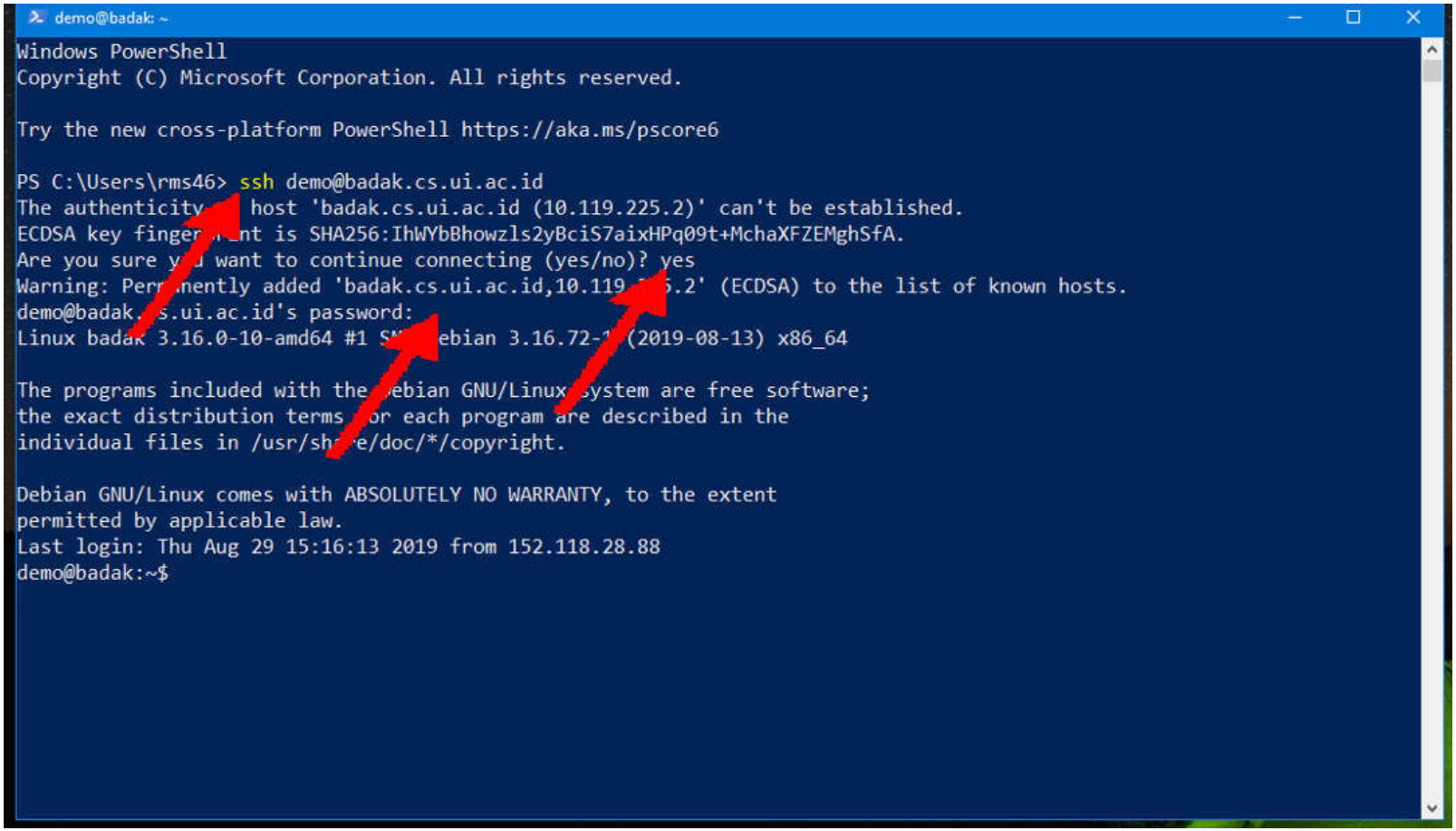


Figure: SSH using PowerShell (Windows 10)

Program Example (Week 00)

```
$ cat c-program-example.c
/* (c) 2016-2019 Rahmat M. Samik-Ibrhaim
 * REV03 Fri Jan 25 18:56:46 WIB 2019
 * REV02 Mon Aug 27 18:17:11 WIB 2018
 * REV01 Sun Aug 20 15:01:12 WIB 2017
 * START Fri Jan 01 00:00:00 WIB 2016
 * This is a free software.
 * To compile:
 * $ gcc -o c-program-example c-program-example.c
 * To execute:
 * $ ./c-program-example
 */

#include <stdio.h>

void main() {
   printf("Hello World!\n");
}
```

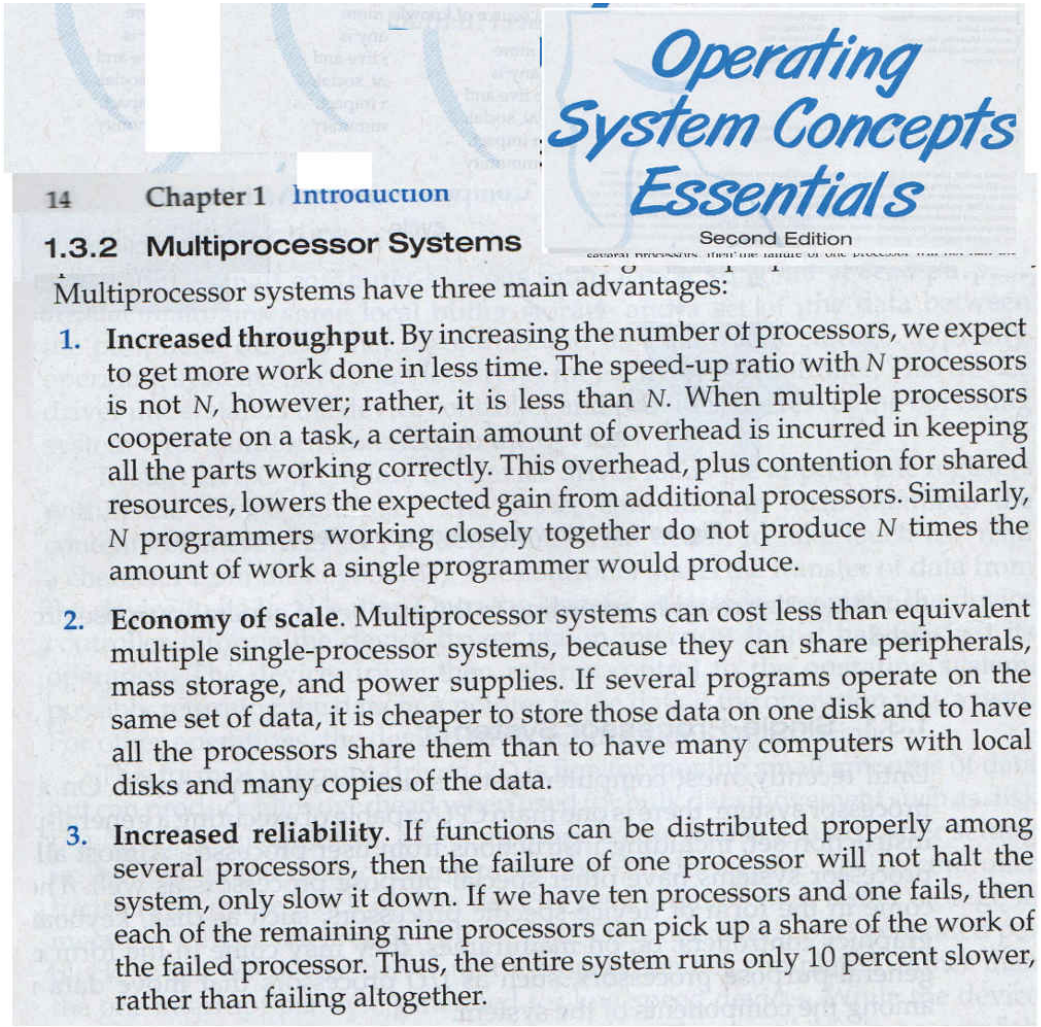
Makefile

```
$ cat Makefile

# (c) 2016-2017 Rahmat M. Samik-Ibrahim
# REV01 Tue Aug 22 14:45:14 WIB 2017
# START Fri Jan 01 00:00:00 WIB 2016
# This is a free Makefile configuration.
# Just run:
# % make

ALL:  c-program-example

c-program-example: c-program-example.c
   gcc -o c-program-example c-program-example.c

clean:
   rm -f c-program-example
```

Week 00: Demo Directory

```
demo@badak:~/mydemo/W00-demos$ PS1="$ "
$ ls -al
total 1080
drwxr-xr-x  2 demo demo   4096 Jan 30 17:35 .
drwx------ 14 demo demo   4096 Jan 30 17:35 ..
-rw-r--r--  1 demo demo   1637 Jan 30 17:35 1-READ-THIS-FIRST.txt
-rw-r--r--  1 demo demo    930 Jan 30 17:35 c-program-example.c
-rw-r--r--  1 demo demo    406 Jan 30 17:35 .head
-rw-r--r--  1 demo demo    376 Jan 30 17:35 Makefile
-rw-r--r--  1 demo demo 516465 Jan 30 17:35 QR-Code.docx
-rw-r--r--  1 demo demo 238225 Jan 30 17:35 QR-Code.pdf
-rw-r--r--  1 demo demo 317401 Jan 30 17:35 QR-Code.png
$ make
gcc -o c-program-example c-program-example.c
$ ./c-program-example
Hello World!
$ ls -F
1-READ-THIS-FIRST.txt  c-program-example*  c-program-example.c Makefile  QR-Code.docx  QR-Code.pdf
      QR-Code.png
$ make clean
rm -f c-program-example
$ ls -F
1-READ-THIS-FIRST.txt  c-program-example.c  Makefile  QR-Code.docx  QR-Code.pdf  QR-Code.png
$
```

Week 00: Problem Example (from OSC2e)



Operating System Concepts Essentials
Second Edition

14 Chapter 1 Introduction

1.3.2 Multiprocessor Systems

Multiprocessor systems have three main advantages:

1. **Increased throughput**. By increasing the number of processors, we expect to get more work done in less time. The speed-up ratio with N processors is not N, however; rather, it is less than N. When multiple processors cooperate on a task, a certain amount of overhead is incurred in keeping all the parts working correctly. This overhead, plus contention for shared resources, lowers the expected gain from additional processors. Similarly, N programmers working closely together do not produce N times the amount of work a single programmer would produce.
2. **Economy of scale**. Multiprocessor systems can cost less than equivalent multiple single-processor systems, because they can share peripherals, mass storage, and power supplies. If several programs operate on the same set of data, it is cheaper to store those data on one disk and to have all the processors share them than to have many computers with local disks and many copies of the data.
3. **Increased reliability**. If functions can be distributed properly among several processors, then the failure of one processor will not halt the system, only slow it down. If we have ten processors and one fails, then each of the remaining nine processors can pick up a share of the work of the failed processor. Thus, the entire system runs only 10 percent slower, rather than failing altogether.

Figure: **T / F** The advantages of a multiprocessor system include: increased throughput, economy of scale, and increased reliability (Week00 2016-1).

TIPS (1)

- For any administrative issues, contact SEKRE at building B, 2^{nd} floor – especially for absences, illness, sick letters, follow-up exams, etc. Please do not contact the **Lecturer** (RMS).
- Please complete the follow-up / paper work within 6 working days (RMS).
- Prepare the weekly MEMO as completely as possible. You should have mastered the material at the beginning of the week (RMS).
- Study the Operating System Concept book which deals with the material will be discussed that week (MIM). Make a summary of material in your Memo (IP).
- You should understand every single problem of the past examinations. Write down all hints in your "**MEMO**" (MHP).
- You are allowed to bring up to 6 sheets of MEMOs for the midterm (UTS) and up to 5 sheets of MEMOs for the final term (UAS) (RMS).
- You should understand every single line of the "**DEMOS**" (MHP).

TIPS (2)

- You should ask **the lecturer** or anyone, anything you do not understand (TA).

TIPS (3)

- TBA.

Special Thanks

Special thanks for the early version of this writing to:

Anisha Inas Izdihar (AII), Benedictus Alvin (BA), Ibnu Sofian Firdaus (ISF), Irmanpen Panjaitan (IP), Ivana Irene Thomas (IIT), Michael Giorgio Wirawan (MGW), Muhammad Afkar (MA), Muhammad Hanif Pratama (MHP), Muhammad Iqbal Mahendra (MIM), M. Ikhsan Kurniawan (MIK), Nixi Sendya Putri (NSP), Raihan Mahendra Sutanto (RM), Rizki Leonardo (RL), Shavira Adeva (SA), Stefan Mayer Sianturi (SMS), Thrisnadevany Amalia (TA), Zhelia Alifa (ZA);

See also `https://rms46.vlsm.org/2/221.pdf`.

Week 00: Summary

- What is an Operating Systems?
  - Definition: Resource Allocator & Control Program.
  - Why taking an Operating System class?
- Computer Organization Review
- The Manager Set
  - Process Manager, Memory Manager, I/O Manager, Storage Manager.
- Security and Protection
- Virtualization
  - Hypervisor type 0, 1, 2
  - Paravirtualization, Emulators, Containers.
  - VCPU: Virtual CPU
  - Virtualization Implementation:
    - Trap-and-Emulate mode
    - Binary Translation mode

Week 00: Check List

- ☐ Starting **Week 01**: TABULA RASA is not accepted anymore!
- ☐ Find/copy this document from `https://os.vlsm.org/`
- ☐ Find/read a recent/decent OS Book and map it to **OSC10**.
- ☐ Using your **SSO** account, login to `badak.cs.ui.ac.id` via `kawung.cs.ui.ac.id`.
- ☐ Check folder `badak:///extra/Demos/Week00/`
  - ☐ Try to copy and compile `c-program-example.c`.
- ☐ QR Code: (Eg) "`OS191 X 1253755125 demo Demo Suremo`"
- ☐ Write "Memo Week00" + your QRC.
- ☐ **How to improve this document?**

Week 00 Overview I: Topics[1]

- Role and purpose of the operating system
- Functionality of a typical operating system
- Mechanisms to support client-server models, hand-held devices
- Design issues (efficiency, robustness, flexibility, portability, security, compatibility)
- Influences of security, networking, multimedia, windowing systems
- Structuring methods (monolithic, layered, modular, micro-kernel models)
- Abstractions, processes, and resources
- Concepts of application program interfaces (APIs)
- The evolution of hardware/software techniques and application needs
- Device organization
- Interrupts: methods and implementations
- Concept of user/system state and protection, transition to kernel mode

[1] Source: ACM IEEE CS Curricula 2013

Week 00 Overview I: Learning Outcomes (1)[1]

- Explain the objectives and functions of modern operating systems [Familiarity]
- Analyze the tradeoffs inherent in operating system design [Usage]
- Describe the functions of a contemporary operating system with respect to convenience, efficiency, and the ability to evolve. [Familiarity]
- Discuss networked, client-server, distributed operating systems and how they differ from single user operating systems. [Familiarity]
- Identify potential threats to operating systems and the security features design to guard against them. [Familiarity]
- Explain the concept of a logical layer. [Familiarity]

[1] Source: ACM IEEE CS Curricula 2013

Week 00 Overview I: Learning Outcomes (2)[1]

- Explain the benefits of building abstract layers in hierarchical fashion. [Familiarity]
- Describe the value of APIs and middleware. [Assessment]
- Describe how computing resources are used by application software and managed by system software. [Familiarity]
- Contrast kernel and user mode in an operating system. [Usage]
- Discuss the advantages and disadvantages of using interrupt processing. [Familiarity]
- Explain the use of a device list and driver I/O queue. [Familiarity]

[1] Source: ACM IEEE CS Curricula 2013

Week 01 Overview II: Topics[1]

- Types of virtualization (including Hardware/Software, OS, Server, Service, Network)
- Paging and virtual memory
- Virtual file systems
- Hypervisors
- Portable and cost of virtualization; emulation vs. isolation
- Cloud services: IAAS, PAAS and Platform APIs, SAAS
- Introduction to Scripting and REGEX.

[1] Source: ACM IEEE CS Curricula 2013

Week 01 Overview II: Learning Outcomes[1]

- Explain the concept of virtual memory and how it is realized in hardware and software. [Familiarity]
- Discuss hypervisors and the need for them in conjunction with different types of hypervisors. [Usage]
- Differentiate emulation and isolation. [Familiarity]
- Evaluate virtualization trade-offs. [Assessment]
- Discuss the importance of elasticity and resource management in cloud computing. [Familiarity]
- Explain the advantages and disadvantages of using virtualized infrastructure. [Familiarity]

[1] Source: ACM IEEE CS Curricula 2013

Week 02 Security & Protection: Topics[1]

- Overview of system security
- Policy/mechanism separation
- Security methods and devices
- Protection, access control, and authentication
- Backups

[1] Source: ACM IEEE CS Curricula 2013

Week 02 Security & Protection: Learning Outcomes[1]

- Articulate the need for protection and security in an OS (cross-reference IAS/Security Architecture and Systems Administration/Investigating Operating Systems Security for various systems). [Assessment]
- Summarize the features and limitations of an operating system used to provide protection and security [Familiarity]
- Explain the mechanisms available in an OS to control access to resources [Familiarity]
- Carry out simple system administration tasks according to a security policy, for example creating accounts, setting permissions, applying patches, and arranging for regular backups [Usage]

[1] Source: ACM IEEE CS Curricula 2013

Week 03 File System & FUSE: Topics[1]

- Files: data, metadata, operations, organization, buffering, sequential, nonsequential
- Directories: contents and structure
- File systems: partitioning, mount/unmount, virtual file systems
- Standard implementation techniques
- Memory-mapped files
- Special-purpose file systems
- Naming, searching, access, backups
- Journaling and log-structured file systems

[1] Source: ACM IEEE CS Curricula 2013

Week 03 File System & FUSE: Learning Outcomes[1]

- Describe the choices to be made in designing file systems. [Familiarity]
- Compare and contrast different approaches to file organization, recognizing the strengths and weaknesses of each. [Usage]
- Summarize how hardware developments have led to changes in the priorities for the design and the management of file systems. [Familiarity]
- Summarize the use of journaling and how log-structured file systems enhance fault tolerance. [Familiarity]

[1] Source: ACM IEEE CS Curricula 2013

Week 04 Addressing: Topics[1]

- Bits, bytes, and words
- Numeric data representation and number bases
- Representation of records and arrays

[1] Source: ACM IEEE CS Curricula 2013

Week 04 Addressing: Learning Outcomes[1]

- Explain why everything is data, including instructions, in computers. [Familiarity]
- Explain the reasons for using alternative formats to represent numerical data. [Familiarity]
- Describe the internal representation of non-numeric data, such as characters, strings, records, and arrays. [Familiarity]

[1] Source: ACM IEEE CS Curricula 2013

Week 05 Virtual Memory: Topics[1]

- Review of physical memory and memory management hardware
- Virtual Memory
- Caching
- Memory Allocation
- Memory Performance
- Working sets and thrashing

[1] Source: ACM IEEE CS Curricula 2013

Week 05 Virtual Memory: Learning Outcomes[1]

- Explain memory hierarchy and cost-performance trade-offs. [Familiarity]
- Summarize the principles of virtual memory as applied to caching and paging. [Familiarity]
- Describe the reason for and use of cache memory (performance and proximity, different dimension of how caches complicate isolation and VM abstraction). [Familiarity]
- Defend the different ways of allocating memory to tasks, citing the relative merits of each. [Assessment]
- Evaluate the trade-offs in terms of memory size (main memory, cache memory, auxiliary memory) and processor speed. [Assessment]
- Discuss the concept of thrashing, both in terms of the reasons it occurs and the techniques used to recognize and manage the problem. [Familiarity]

[1] Source: ACM IEEE CS Curricula 2013

Week 06 Concurrency: Topics[1]

- States and state diagrams
- Structures (ready list, process control blocks, and so forth)
- Dispatching and context switching
- The role of interrupts
- Managing atomic access to OS objects
- Implementing synchronization primitives
- Multiprocessor issues (spin-locks, reentrancy)

[1] Source: ACM IEEE CS Curricula 2013

Week 06 Concurrency: Learning Outcomes (1)[1]

- Describe the need for concurrency within the framework of an operating system. [Familiarity]
- Demonstrate the potential run-time problems arising from the concurrent operation of many separate tasks. [Usage]
- Summarize the range of mechanisms that can be employed at the operating system level to realize concurrent systems and describe the benefits of each. [Familiarity]
- Explain the different states that a task may pass through and the data structures needed to support the management of many tasks. [Familiarity]

[1] Source: ACM IEEE CS Curricula 2013

Week 06 Concurrency: Learning Outcomes (2)[1]

- Summarize techniques for achieving synchronization in an operating system (e.g., describe how to implement a semaphore using OS primitives). [Familiarity]
- Describe reasons for using interrupts, dispatching, and context switching to support concurrency in an operating system. [Familiarity]
- Create state and transition diagrams for simple problem domains. [Usage]

[1] Source: ACM IEEE CS Curricula 2013

Week 07 Synchronization & Deadlock: Topics[1]

- Shared Memory and Critical Section
- Consistency, and its role in programming language guarantees for data-race-free programs
- Message passing: PtPo vs Multicast, Blocking vs non-blocking, buffering.

[1] Source: ACM IEEE CS Curricula 2013

Week 07 Synchronization & Deadlock: Learning Outcomes[1]

- Use mutual exclusion to avoid a given race condition. [Usage]
- Give an example of an ordering of accesses among concurrent activities (e.g., program with a data race) that is not sequentially consistent. [Familiarity]
- Use semaphores to block threads [Usage]

[1] Source: ACM IEEE CS Curricula 2013

Week 08 Scheduling: Topics[1]

- Preemptive and non-preemptive scheduling
- Schedulers and policies
- Processes and threads
- Deadlines and real-time issues

[1] Source: ACM IEEE CS Curricula 2013

Week 08 Scheduling: Learning Outcomes[1]

- Compare and contrast the common algorithms used for both preemptive and non-preemptive scheduling of tasks in operating systems, such as priority, performance comparison, and fair-share schemes. [Usage]
- Describe relationships between scheduling algorithms and application domains. [Familiarity]
- Discuss the types of processor scheduling such as short-term, medium-term, long-term, and I/O. [Familiarity]
- Describe the difference between processes and threads. [Usage]
- Compare and contrast static and dynamic approaches to real-time scheduling. [Usage]
- Discuss the need for preemption and deadline scheduling. [Familiarity]
- Identify ways that the logic embodied in scheduling algorithms are applicable to other domains, such as disk I/O, network scheduling, project scheduling, and problems beyond computing. [Usage]

[1] Source: ACM IEEE CS Curricula 2013

Week 09 Storage, Firmware, Bootloader, & Systemd: Topics[1]

- Storage
- Storage Arrays
- BIOS
- Loader
- Systemd

[1] Source: ACM IEEE CS Curricula 2013

Week 09 Storage, Firmware, Bootloader, & Systemd: Learning Outcomes[1]

- Storage [Usage]
- Storage Arrays [Usage]
- BIOS [Usage]
- Loader [Usage]
- Systemd [Usage]

[1] Source: ACM IEEE CS Curricula 2013

Week 10 I/O & Programming: Topics[1]

- Characteristics of serial and parallel devices
- Abstracting device differences
- Buffering strategies
- Direct memory access
- Recovery from failures
- I/O Programming
- Network Programming

[1] Source: ACM IEEE CS Curricula 2013

Week 10 I/O & Programming: Learning Outcomes[1]

- Explain the key difference between serial and parallel devices and identify the conditions in which each is appropriate. [Familiarity]
- Identify the relationship between the physical hardware and the virtual devices maintained by the operating system. [Usage]
- Explain buffering and describe strategies for implementing it. [Familiarity]
- Differentiate the mechanisms used in interfacing a range of devices (including hand-held devices, networks, multimedia) to a computer and explain the implications of these for the design of an operating system. [Usage]
- Describe the advantages and disadvantages of direct memory access and discuss the circumstances in which its use is warranted. [Usage]
- Identify the requirements for failure recovery. [Familiarity]
- Implement a simple device driver for a range of possible devices. [Usage]
- I/O Programming [Usage]
- Network Programming [Usage]

The End

- ☐ This is the end of the presentation.
- ☒ This is the end of the presentation.
- This is the end of the presentation.