Operating Systems

Nipun Batra
Aug 2, 2018

Question: What all OS’ have you used?

Question: What all OS' have you used?

Any major change in recent OS?
Video: Linux , Windows

Why Study OS?

- “Systems” bucket - important branch of CS
- Use it everyday - Gratitude towards those who made our lives easier!
- Application of DS, Algorithms
  - Heaps, Stacks,…
- Placements?! :)

What is an OS?

Hardware

Applications

Users

Discussion - What happens if I start Pages/ Word in the middle of this presentation?

Discussion - What happens if I start Pages/ Word in the middle of this presentation?

1. Next time I open Keynote/Adobe Reader - where does this presentation resume from -> Store processing and memory state of current program
2. I click on Pages Icon. How does my system know I clicked? -> Need to interpret trackpad events
3. Ok. System knows that Pages needs to be started -> where is Pages stored?
4. Pages started - where does the program reside now?
5. I type something in it -> how does it show on the monitor? where is the data that I typed stored? What happens when I save it?
6. Let's start a C program now. Let's write it in an editor. What happens if I open same file in two locations?

Discussion - What happens if I start Pages/ Word in the middle of this presentation?

1. Next time I open Keynote/Adobe Reader - where does this presentation resume from -> Store processing and memory state of current program
2. I click on Pages Icon. How does my system know I clicked? -> Need to interpret trackpad events
3. Ok. System knows that Pages needs to be started -> where is Pages stored?
4. Pages started - where does the program reside now?
5. I type something in it -> how does it show on the monitor? where is the data that I typed stored? What happens when I save it?
6. Let's start a C program now. Let's write it in an editor. What happens if I open same file in two locations?

Discussion - What happens if I start Pages/ Word in the middle of this presentation?

OS manages running multiple programs

2. I click on Pages Icon. How does my system know I clicked? -> Need to interpret trackpad events
3. Ok. System knows that Pages needs to be started -> where is Pages stored?
4. Pages started - where does the program reside now?
5. I type something in it -> how does it show on the monitor? where is the data that I typed stored? What happens when I save it?
6. Let's start a C program now. Let's write it in an editor. What happens if I open same file in two locations?

Discussion - What happens if I start Pages/ Word in the middle of this presentation?

OS manages running multiple programs

OS interfaces with hardware using easy interface

3. Ok. System knows that Pages needs to be started -> where is Pages stored?
4. Pages started - where does the program reside now?
5. I type something in it -> how does it show on the monitor? where is the data that I typed stored? What happens when I save it?
6. Let's start a C program now. Let's write it in an editor. What happens if I open same file in two locations?

Discussion - What happens if I start Pages/ Word in the middle of this presentation?

OS manages running multiple programs

OS interfaces with hardware using easy interface

OS transforms programs to processes

4. Pages started - where does the program reside now?
5. I type something in it -> how does it show on the monitor? where is the data that I typed stored? What happens when I save it?
6. Let's start a C program now. Let's write it in an editor. What happens if I open same file in two locations?

OS manages resources - CPU, Memory, Disk, Peripherals

OS manages running multiple programs

OS interfaces with hardware using easy interface

OS transforms programs to processes

4. Pages started - where does the program reside now?
5. I type something in it -> how does it show on the monitor? where is the data that I typed stored? What happens when I save it?
6. Let's start a C program now. Let's write it in an editor. What happens if I open same file in two locations?

Logistics

Course Website has all details
https://nipunbatra.github.io/teaching/os-fall-18/index.html

3 Easy Steps / 3 Parts of The Course

3 Easy Steps / 3 Parts of The Course

1. Virtualisation : Physical resource (CPU, disk, memory) -> virtual resource

3 Easy Steps / 3 Parts of The Course

1. Virtualisation : Physical resource (CPU, disk, memory) -> virtual resource
    1. We saw in previous example, multiple:

3 Easy Steps / 3 Parts of The Course

1. Virtualisation : Physical resource (CPU, disk, memory) -> virtual resource
    1. We saw in previous example, multiple:
        1. Programs running simultaneously, each thinks they have CPU to themselves

3 Easy Steps / 3 Parts of The Course

1. Virtualisation : Physical resource (CPU, disk, memory) -> virtual resource
    1. We saw in previous example, multiple:
        1. Programs running simultaneously, each thinks they have CPU to themselves
        2. Each thinks they have memory

3 Easy Steps / 3 Parts of The Course

1. Virtualisation : Physical resource (CPU, disk, memory) -> virtual resource
    1. We saw in previous example, multiple:
        1. Programs running simultaneously, each thinks they have CPU to themselves
        2. Each thinks they have memory
2. Concurrency : Running multiple things at once

3 Easy Steps / 3 Parts of The Course

1. Virtualisation : Physical resource (CPU, disk, memory) -> virtual resource
    1. We saw in previous example, multiple:
        1. Programs running simultaneously, each thinks they have CPU to themselves
        2. Each thinks they have memory
2. Concurrency : Running multiple things at once
    1. Can cause problems!

3 Easy Steps / 3 Parts of The Course

1. Virtualisation : Physical resource (CPU, disk, memory) -> virtual resource
   1. We saw in previous example, multiple:
      1. Programs running simultaneously, each thinks they have CPU to themselves
      2. Each thinks they have memory
2. Concurrency : Running multiple things at once
   1. Can cause problems!
3. Persistence : Store data permanently

CPU Virtualisation Demo

CPU Virtualisation Demo

1. Run cpu-virtual.py

CPU Virtualisation Demo

1. Run cpu-virtual.py
2. See the output of **ps** and **top**

CPU Virtualisation Demo

1. Run cpu-virtual.py
2. See the output of **ps** and **top**
3. See the activity monitor

CPU Virtualisation Demo

1. Run cpu-virtual.py
2. See the output of **ps** and **top**
3. See the activity monitor
4. **OS-Fun** Can you do something to make top show a better name than just Python3.6? Can you do it for a C program?

Memory Virtualisation Demo

Memory Virtualisation Demo

1. Run mem.c

Memory Virtualisation Demo

1. Run mem.c
2. Wait why different addresses?!

Memory Virtualisation Demo

1. Run mem.c
2. Wait why different addresses?!
    1. Address space randomisation!

Memory Virtualisation Demo

1. Run mem.c
2. Wait why different addresses?!
    1. Address space randomisation!
    2. Re-run with it disabled

Memory Virtualisation Demo

1. Run mem.c
2. Wait why different addresses?!
    1. Address space randomisation!
    2. Re-run with it disabled
    3. **OS-Fun:** Read about ASLR - why does it make sense?

Memory Virtualisation Demo

1. Run mem.c
2. Wait why different addresses?!
    1. Address space randomisation!
    2. Re-run with it disabled
    3. **OS-Fun:** Read about ASLR - why does it make sense?
3. See the activity monitor

Memory Virtualisation Demo

1. Run mem.c
2. Wait why different addresses?!
   1. Address space randomisation!
   2. Re-run with it disabled
   3. **OS-Fun:** Read about ASLR - why does it make sense?
3. See the activity monitor
4. Run vmmap

Concurrency

$\neq$

Concurrency

1. We discussed previously how OS juggles between multiple processes

$\neq$

Concurrency

1. We discussed previously how OS juggles between multiple processes
2. Run python thread.py #loops

$\neq$

Concurrency

1. We discussed previously how OS juggles between multiple processes
2. Run python thread.py #loops
    1. Different answers?

$\neq$

Concurrency

1. We discussed previously how OS juggles between multiple processes
2. Run python thread.py #loops
    1. Different answers?
    2. Counter $\neq$ N * # Loops. Why?

Concurrency

1. We discussed previously how OS juggles between multiple processes
2. Run python thread.py #loops
    1. Different answers?
    2. Counter $\neq$ N * # Loops. Why?
    3. Update operation:

Concurrency

1. We discussed previously how OS juggles between multiple processes
2. Run python thread.py #loops
    1. Different answers?
    2. Counter $\neq$ N * # Loops. Why?
    3. Update operation:
        1. Load Counter into register

Concurrency

1. We discussed previously how OS juggles between multiple processes
2. Run python thread.py #loops
    1. Different answers?
    2. Counter $\neq$ N * # Loops. Why?
    3. Update operation:
        1. Load Counter into register
        2. Update Counter

Concurrency

1. We discussed previously how OS juggles between multiple processes
2. Run python thread.py #loops
   1. Different answers?
   2. Counter $\neq$ N * # Loops. Why?
   3. Update operation:
      1. Load Counter into register
      2. Update Counter
      3. Store Counter

Concurrency

1. We discussed previously how OS juggles between multiple processes
2. Run python thread.py #loops
    1. Different answers?
    2. Counter $\neq$ N * # Loops. Why?
    3. Update operation:
        1. Load Counter into register
        2. Update Counter
        3. Store Counter
    4. Non-atomic update!

Concurrency

1. We discussed previously how OS juggles between multiple processes
2. Run python thread.py #loops
    1. Different answers?
    2. Counter $\neq$ N * # Loops. Why?
    3. Update operation:
        1. Load Counter into register
        2. Update Counter
        3. Store Counter
    4. Non-atomic update!

Filesystem

Design Goals

Design Goals

1. High performance -> Minimize OS overheads

Design Goals

1. High performance -> Minimize OS overheads
    1. Extra memory

Design Goals

1. High performance -> Minimize OS overheads
    1. Extra memory
    2. Extra CPU

Design Goals

1. High performance -> Minimize OS overheads
    1. Extra memory
    2. Extra CPU
    3. Extra disk

Design Goals

1. High performance -> Minimize OS overheads
   1. Extra memory
   2. Extra CPU
   3. Extra disk
2. Protecting applications from one harming another and the OS -> Isolation

Design Goals

1. High performance -> Minimize OS overheads
    1. Extra memory
    2. Extra CPU
    3. Extra disk
2. Protecting applications from one harming another and the OS -> Isolation
3. Reliability

Design Goals

1. High performance -> Minimize OS overheads
   1. Extra memory
   2. Extra CPU
   3. Extra disk
2. Protecting applications from one harming another and the OS -> Isolation
3. Reliability
   1. Imagine sitting in a flight and the OS crashing!

Design Goals

1. High performance -> Minimize OS overheads
    1. Extra memory
    2. Extra CPU
    3. Extra disk
2. Protecting applications from one harming another and the OS -> Isolation
3. Reliability
    1. Imagine sitting in a flight and the OS crashing!
    2. Or, dispensing cash in an ATM and the OS crashing!

Design Goals

1. High performance -> Minimize OS overheads
    1. Extra memory
    2. Extra CPU
    3. Extra disk
2. Protecting applications from one harming another and the OS -> Isolation
3. Reliability
    1. Imagine sitting in a flight and the OS crashing!
    2. Or, dispensing cash in an ATM and the OS crashing!
    3. Or, the MRI scan machine OS reboots on its own!

Design Goals

1. High performance -> Minimize OS overheads
    1. Extra memory
    2. Extra CPU
    3. Extra disk
2. Protecting applications from one harming another and the OS -> Isolation
3. Reliability
    1. Imagine sitting in a flight and the OS crashing!
    2. Or, dispensing cash in an ATM and the OS crashing!
    3. Or, the MRI scan machine OS reboots on its own!
4. Energy efficiency (esp. for mobile systems!)