operating system fundamentals

Session 3

قولنا قبل كده ان ال computer h.w ← i/o devices , RAM, CPU

وقولنا ان ال Process بتستخدم ال resources دي عشان تشتغل في بيحصلها

execution ع ال CPU , loading في ال memory وبتعمل عمليات operations مع ال i/o devices.

و طبعا ال Process هتتعامل مع ال resources دي فأكيد فيه طريقة اتعامل management

هنبقى نقول هنا management لحاجات كتيرة أوي

(Process, main memory, file system, i/o devices, secondary storage)

فاحنا بقا هنشرح كل واحدة من دي بالتفصيل

Process management

طب نعرف يعني ايه الأول Process

A process is a program in execution

طب ايه الفرق بين ال process و ال program

ال program هو الفايل المكتوب فيه الكود ال محفوظ بعد كده هيحصله execution

الفايل ده طول ما هو متخزن في الهارد ديسك فهو اسمه program

أول ما تضغط عليه douple click ويحصله loading في الميموري فهو بيبقى Process.

- A process needs certain resources, including CPU, memory, files, and i/o devices, to accomplish its task.

وده قولناه فوق

- The operating system is responsible for the following activities in connection with process management :-

① Process creation and deletion

ال Process creation يعني أول ما تضغط click يبدأ ال os يحجز لل Process

دي مكان في الميموري ويشوف لو محتاجة i/o devices ويحط information عنها

بعدها يحصل كل ده يبدأ يحصل ال loading لل APP من ال hard disk ويحطه في

ال main memory [فيه فرق بين ال creation و ال loading]

ال Process deletion ← ده معناه يحصل لو ال ← Process مثلا خلصت

في بيشيلها من ال memory

② Process suspension and resumption

← إني أعمل Pause لل Process وأرجع أشغلها (ال os المسؤول مش أنا :D)

← وده في حالة ال interrupt

← أو مثلاً إن ال time sharing بتاع ال process خلص فلازم تطلع من البروسيسور

③ Process communication

← ال os بيعمل لل Processes ← synchronization عشان لو فيه كذا Process وعاوزين يستخدموا نفس الحاجة في نفس الوقت (حاجة زي طابعة مثلاً، ال i/o device)

← ال Communication في حالة two processes عاوزين يشيروا data مع بعض

---

main memory management

- memory is a large array of bytes, each with its own address, it is a repository of quickly accessible data shared by the CPU and i/o devices.

← ال main mem هي أكتر مكان ال CPU بيعملها direct access

- main memory is a Volatile storage device, it loses its contents in the case of system failure

← الكهربا قطعت هوب الداتا طارت

- The os is responsible for the following activities in connections with memory management:

① Keep track of which parts of memory are currently being used and by whom

(زي مثال إنك لو فتحت ال task manager هتلاقي ال os بيقولك كل الموقعه)

② Decide which Proccesses to load when memory space becomes available

← لو البروسيسور فيها Process شغالة وفي Process تانية وعطل ال os يشغلها فبيعمل حاجة اسمها Virtual mem ويعمل swap بينهم

③ allocate and de-allocate memory space as needed

← لو فيه Process شغالة وإحتاجت مساحة معينة في ال os هيعمل allocation لل space ده وبعدما ال Process تخلص بيعملها deallocation

File management

- A file is a collection of related information defined by its creator.
  Commonly, files represent programs and data
  ← الفايل هو مجموعة من الداتا محطوطة مع بعض بفورمات معين

- The os is responsible for the following activities in connections with file management:

① file creation and deletion
← يعني أنا لما حاول اعمل فايل أو امسح فايل ال فعليا بيعمل كده هو ال os

② direcory creation and deletion

③ support of primitives for manipulating files & directori
← ال os هو ال بتوفر لي ال tools ال تخليني أعدل لو privilige ليا كـ يوزر

④ mapping files onto secondary storage (hard disk)
← الفايل متخزن في ال hard disk في مجموعة من ال sectors مش ورا بعض
عليه process فهيتعرض في الصورة بشكل byte المفروض فكده
عندي two structers مختلفين فال هيعمل mapping بينهم هو ال os

⑤ file backup on stable (non-volatile) storage media

i/o system management

- os hide Particularities of i/o devices

← ال os بيخفي تفاصيل ال i/o devices عن اليوزر (APP Program)، لأن أي instructions of i/o device
تعتبر Privilege instruction ← معناها: مينفعش تتنفذ إلا لما تكون محجوزة عن طريق ال os
طب ال i/o device هيعرف منين إنها محجوزة عن طريق ال os ؟! فيه mode bits
← عن طريق ال mode هيشوف هل هو user mode ولا monitor mode

- Device drivers
  input : retrieve block of data
  output : hardware instructions of controller

هو software بين ال os وبين ال device controller

← ال os لما بيعمل detection ل device فبعض الأحيان مبيعرفش detect نوع ال device ده فبالتالي ال os مش هيعرف يتكلم مع ال device controller
فـ محتاج ال device driver في النص بينهم

← لو ال os قدر يتكلم مع ال device controller على طول ده معناه إن ~~device controller~~ الـ os جوه ال device driver الخاص بال device ده

- i/o Subsystem
  ↳ memory managent : spooling & . Drivers for specific hardware

secondary-storage management

- since main memory (primary storage) is volatile and too small to accommodate all data and programs permanently, the computer system must provide secondary storage to back up main memory.
- most modern computer systems use disks as the principle on-line storage medium, for both programs and data.
- The os is responsible for the following ------- with disk mangem

① Free space management
② storage allo cation

← يعني الهارد بتاعك ظاهرلك partitions معينة وجواها files لكن physical مش كده بس ال ظاهر لنا ده بالشكل ده هو ال os يعني أنت لو فتحت الهارد نفسه فيزيكالي فتقدرش تحدد ال D, C

③ disk scheduling

← العملية ال swapping بين ال main memory & virtual memory هو ال os

Networking (Distributed systems)

- A distributed system is a collection Processor that don't share memory or a clock.
  - ↳ each Processor has it's own local memory.

- The processors in the system are connected through a Communication network.
  - ↳ Communication takes place using a protocol.

- A distributed system provides user access to various system resources

- Access to a shared resource allow
  1. Computation speed-up
  2. increased data availability
  3. Enhanced reliability

(كل الحاجات دي شرحناها قبل كده فمش هنشرحها تاني تاني :D)

Protection system

- Protection refers to a mechanism for controlling access by programs or users to system resources

فيه أكتر من process شغالة مع بعض فـ إزاي نحميهم من بعض وإزاي نحمي الـ devices i/o منهم

- The protection mechanism must:
  1. distinguish between authorized and unauthorized usage.
     Authentication هل اليوزر ع الـ system ده ليه صلاحيات ولا لا
  2. specify the controls to be imposed

Command - interpreter system.

- many Commands are given to the os by Control statement which deal with:

(Process creation & management , main memory management, secondary storage management, file system access , Protection networking).

طب هو ايه هو اصلاً ؟

The Program that reads and interprets control statements is called variously:

- ex: (Command - line interpreter - shell in unix / linux, وحاجات تانية.

بترجم ال command الي بتحولها لل instructions لل os وبوفر ال interface لليوزر

زي الشاشه اللي قدامك دلوقتي في لينكس وهو ال GUI في ويندوز (يعني هي بتقولي اي لل os)

- its function is to get and execute the next Command - statement

operating system services

- Program execution
  - system capability to load a Program into memory and to run it

- i/o operations
  - since user Programs can't execute i/o operations directly, The os must Provide some means to Perform i/o.

- file system manipulation
  - Program capability to read , write, create, delete files.

(كل ده شرحناه قبل كده برضه)

Operating system services

① Communications

↳ exchange of information between Processes executing either on the same Computer or on different systems tied together by a network.

↳ implemented via shared memory or message Passing

② Error detection

↳ ensure Correct Computing by detecting errors in CPU and memory hardware, in i/o devices, or in user Programs.

System calls

← مجموعة طلبات بتطلبها ال Process من ال os

← كل os بيحدد مجموعة ال system calls ثابتة بعرض مثلاً مجموعة الأوامر الخاصة بال printer، ال os حاططها وكاتبها مثلاً printing فـ بيقول أي جهاز بيتعامل مع ال printer يبعت system call لـ printing.

← لو مثلاً جت قابل etc. وروحت تعمله exection & linud فـ مش هيشتغل ومن ضمن الأسباب إنه مش هيتغلش إن ال sys call ثابتة وبيبقى غير ثابتة الشغل

↓

- system calls Provide the interface between a running Program and the os.

↓

- Generally available as assembly-language instructions → لـ هي مبتتحول إلى أصفار ووحايد

- language defined to replace assemble language for systems Programming allow system calls to be made directly (C, C**, ----)

اللغة هي المسئولة إنها تحول الكود بتاعها إلى ال system call المناسبة وتعمل كده اللغة عن طريق ال Compiler

ج

Copy Program Example

لو عاوز أعمل copy لحاجة دي هستخدم كام system calls ؟!

(1) variable initialization

إن أنت هتشغل الفايل مثلاً من مكان لمكان فأنت لازم تدي ال os الأماكن دي
كأنك بتديله input مثلاً وفي بعض الأحيان مش محتاج أدي ال os أي input
زي مثلاً لو طلبت منه يرجعلي حاجه معينه
وبالمثل أحياناً يحتاج من ال os ← يدينا output زي لو طلبت منه يديني حرف معين وفي بعض
الأحيان مش ستنى منه output

(الخلاصة: أول system calls إما هستنى من input أولاً أو هستنى منها output أولاً)
في ده كده variable initialization

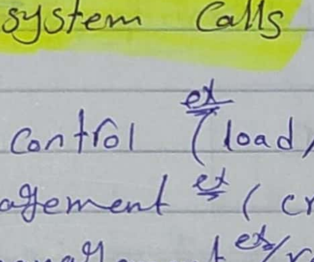
(2) open(file1) . (3) create (file2) (4) read (file1)
(5) write(file2) (6) close(file1) (7) close (file2)

Types of system calls

(1) process control ex (load, end, execute, create, wait time)
(2) file management ex (create, delet, open, close, read, ...)
(3) device management ex (request, release, read, write, get, set attributes)
(4) information maintenance (get/set time & date, get/set process, file)
(5) communications ex ((creat, delet connection), (send, receive message))
(والبقية في السلايدز)

Communication models → الطرق المستخدمة عشان ال processes تتكلم بعض

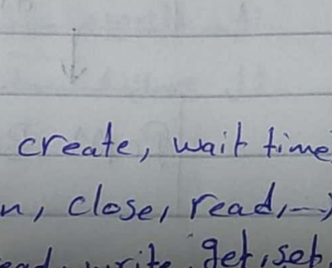
message passing ← الطريقة دي بيدخل فيها ال os



يعني A عاوزه تبعت ل B
في A تروح لل os تقوله عاوزه
أبعت دي ل B في ال os
ياخدها ويبعتها هو ل B
عن طريق system calls

shared memory

لو A عاوز يبعت ل B ↓
هنا A , B هيتفقوا على
مكان لما يحبوا يكلموا بعض
يبعتوا فيه وطبعاً ده بيتم
بموافقة ال os
عشان يمنع أي process
تانيه تلمس المكان ده

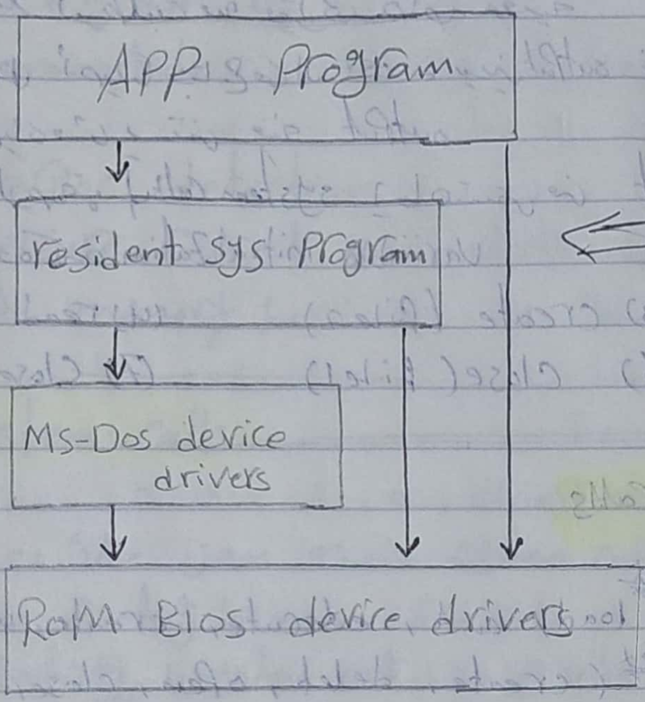


فهم هنا مش بيحتاجوا ل os ولكن دوره
في جزء ال shared mem وبعد فترة أقدر أمسح
المكان ده لكي يتم delet communication

OS system Design structure

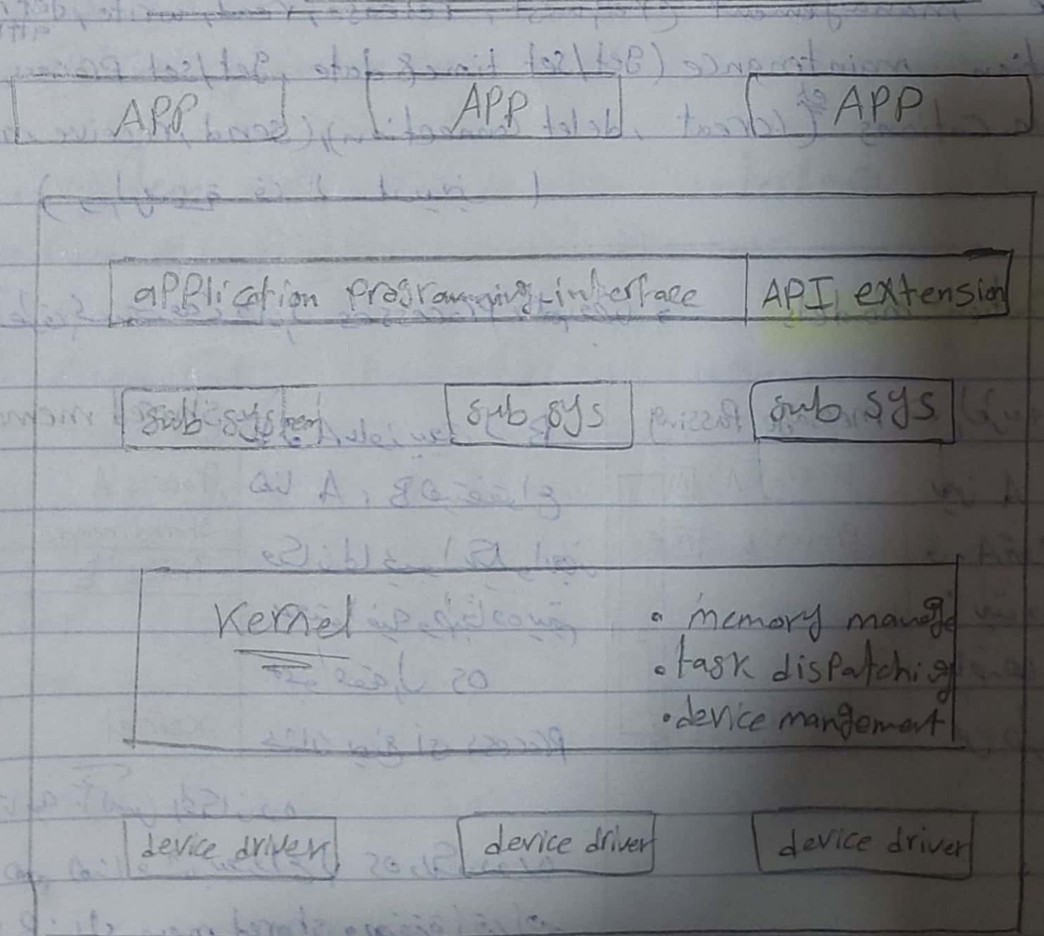
اوصنا شوفنا ال component بتاعت ال os sys طب ازاي
ال component دي بتتركب مع بعض عن طريق

① simple structure   ② layered approach



① [MS-DOS structure = Simple structure]

limitations وفيه



② OS/2 layer structure = layered approach

② layered Approach

كل واحدة فيهم بتتعامل غير مع اللي تحتها بس

- The OS is divided into a number of layers/levels, each built on top of lower layers.

The bottom layer/layer zero → is the hardware
The highest layer/layer N → is the user interface

فـ انت كده layer 1 مثلاً مش شايفه غير layer 0 وهكذا ....

- with modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers

يعني مثلاً لو CPU layer طلب من الـ memory layer عاوزه الـ ip بتاع process معينة فـ الـ memory بتستخدم Algorithm معين، لو مجموعة من الـ developers قرروا يغيروا الـ Algorithm ده، هل من هيأثر على الـ CPU layer؟ الإجابة لأ، لأن التواصل بين الـ two layers عن طريق system calls.

← الـ API مجموعة من الـ libraries بستخدمها عشان أتعامل مع device أو مع software (عشان الـ APP تتكلم مع الـ hw)
← مجموعة من الـ libraries الـ تغلف أعمال الـ system call معين
← موجود في الـ os ده وفي أي hardware

Advantages of layered approach

① modularity → مجموعة من الـ modules المفصولة عن بعض
② Debugging → ده بيغنيني في حالة بعمل test بين الـ device driver مثلاً وبين الـ h.w controller ومفيش أي Error وجيت أعمل test بين الـ kernel والـ device driver وبين الـ controller ولاقيت error فـ كده أكيد الإيرور من الـ kernel
فـ الـ layer ساعدتني في ده.
③ modification

disadvantages

① layering overhead to the system call
← عشان هم كتير ومحتاجين وقت في ترتيب الـ layer كان

Operating system fundamentals

Session 4

اتفقنا إن ال APP وهو موجود على الهارد ديسك بنقول عليه Program
ولما بيبدأ يـ load في ال memory ويحصله execution بنقول عليه Process
واتكلمنا قبل كده ع أنواع ال systems ← Batch, Time shared sys
وقولنا إن ال Batch إن single Process شغالة في الميموري طول الوقت ومفيش في ال memory غيرها
واتكلمنا عن ال multi Programmed ودة فيه أكتر من Process بيحطها load في الميموري وواحدة منهم بس بتتنفذ في ال CPU لحد ما تطلب I/O وهكذا...
واتكلمنا عن مشكلة في Processes تفضل مستنية ومتشتغلش فقولنا هنحل ال Time shared حل المشكلة دي بحيث يكون فيه time slote لكل Process

ملحوظة ← مصطلح ال Job هو نفسه مصطلح ال Process في السلايدز

- Process
  ↳ a program in execution, Process execution must progress in sequential fashion.

- An operating system execute a variety of programs:
  - ⊙ Batch system
    - (simple - multi programmed)
  - ⊙ Time - shared systems

لما ال Process بيحصلها load في ال memory ايه الأجزاء ال بيحصلها load
(المكونة للـ Process)

• Process Contents

① Text Section

• Progam instructions

ده بيكون فيه ال instructions ال المفروض تتنفذ في ال Process دي.

② Program counter

• next instructions

قولنا من ضمن مكونات ال cpu فيه register و ال registers دي فيه منها IR register هو instruction register ← هو register بيتخزن فيه ال instruction ال بتتنفذ دلوقتي

إنما ال Program counter فيه عنوان ال instruction ال عليها الدور في التنفيذ
ده هيفرق لو فيه process بسبب ما حصل interrupt وطلعت برا ال cpu
فمثلاً هي وقفت عند ال instruction رقم (10) دلوقتي لو هترجع تاني لل cpu
هل هتبدأ من الأول ولا تكمل من instruction رقم (11)؟ أكيد هتكمل
هتعرف منين بقا رقم ال instruction من ال Program counter

③ stack

بيتعامل بمبدأ ← last in first out (آخر ما جه عـ أول ش :D)
وال stack ده بيتحط فيه جزء من ال data والجزء التاني في ال Heap
والداتا ف ال stack زي

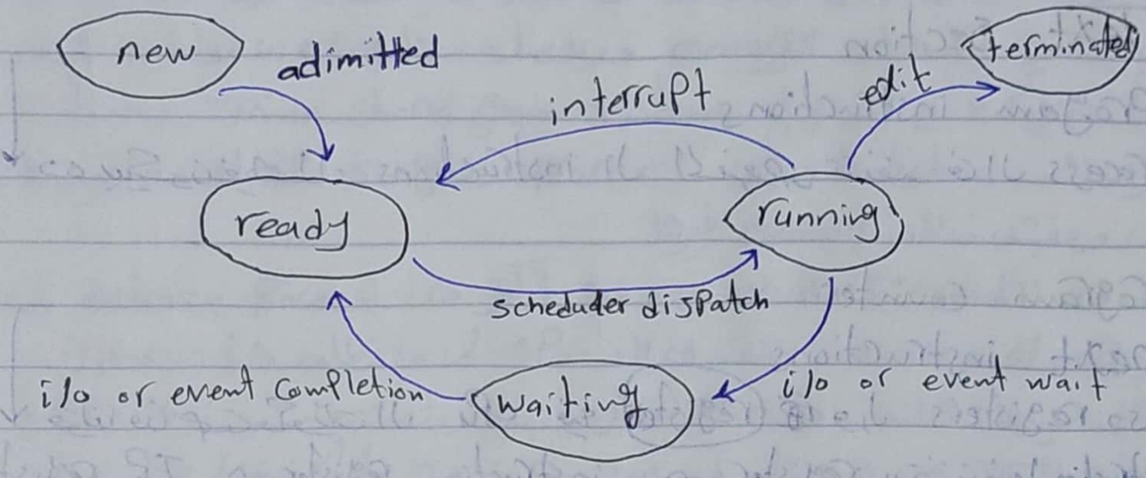
• local variables
• Return addresses
• method Parameters

④ Data Section

• Global Variables

بعد ما شوفنا مكونات ال Process تعالى نشوف ايه هي ال States ال بتعدي بيها / عليها ال Process دي ⟵

Process State



① new ← أول ما ال state ← تكون ال Process عملها (creation)
بس لسه معملهاش load في الميموري ولما عملها load كده بقت ↓
② ready ← إنها تشتغل بعد ما عملها load في الميموري ال os قررت تشغلها ↓
③ running ← لو شغلت يبقى كده بقت (executed)
a ← ④ terminated ← يعني exit و خلصت (finished execution)
b ← في حالة وهي running احتاجت مثلاً i/o فهتطلع من ال cpu
فال os مش هتطلعها يرجعها ready تاني ع طول عشان كده ممكن
تدخل تاني في running ومحتاجة العاوزاه من ال i/o فهيسيبها
⑤ waiting ~~...~~ ولما تخلص ال i/o هيرجعها ready
c ← لو ال time slot تاع ال Process خلصت وال os طلعها من ال cpu
هتحطها في ال ready لأنها مش محتاجه حاجه هي جاهزة أصلاً
d ← لو ال Process عملها s.w/h.w interrupt برضو هيدخلها ready

(a, b, c, d) ← دي الحالات ال طلعت فيها من ال cpu
↘ (terminated, i/o, time slote finished, interrupt)
ready → waiting to be assigned to a Processor.

Process Control Block (PCB)

مجموعة البيانات ال بيخزنها ال OS عن كل process عنده بيخزنها في ال PCB وهو مكان في الميموري

- information associated with each process :↓

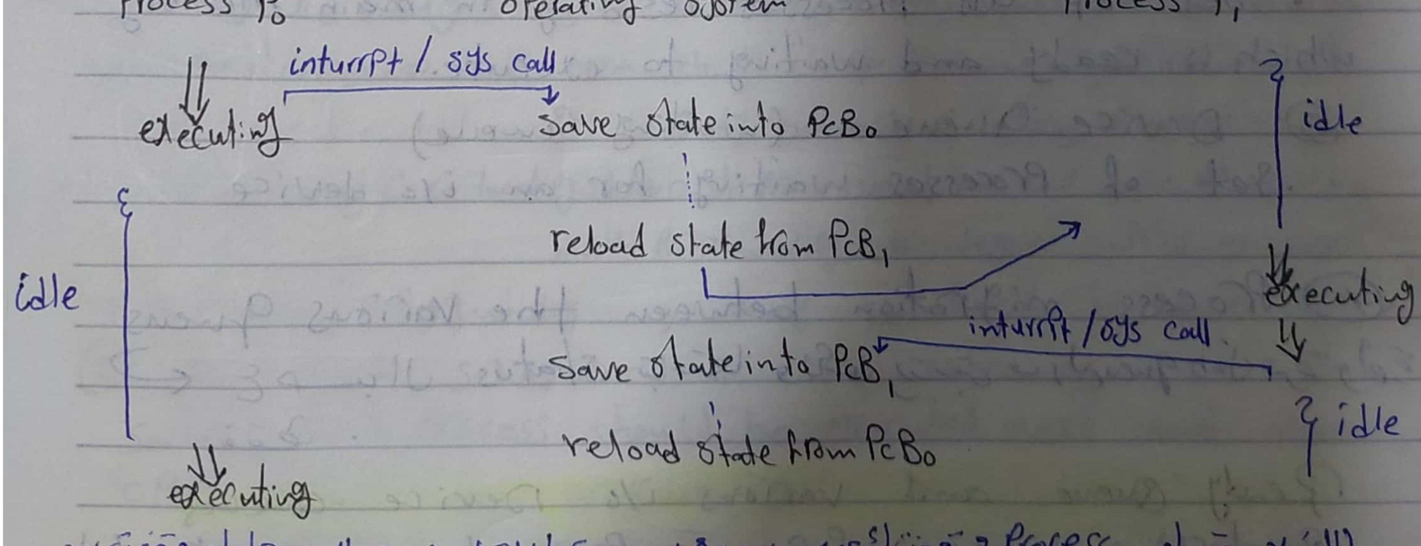
Process state , Program counter , CPU registers ,
CPU scheduling information → (مين عليها الدور تدخل ل CPU) ,
memory - management information → (عشان أمنع ال process تخزن في مكان مش مسموح لها)
accounting information → (بشوف الوقت ال استهلكه ال process في كل state)
i/o status information

General form of PCB

Pointer	Process state
Process number	
Program Counter	
CPU registers	
memory mangement info	
i/o status information	
Accounting information	

⊙ بس ده ميمنعش إن ممكن ال OS يخزن معلومات غير دي عنده

لما ال OS هيغير يطلع process من ع ال CPU ده ال هيحصل ↓



(الخلاصة لو process وقفت لأي سبب بعمل save لحالتها وبروح لل بعدها لو وقفت برده بسيڤ وبـ reload ال بعدها او لو مفيش برجع للأولى وهكذا .)

Process scheduling

→ multi Programming systems
- Some Process executing at all times
- maximize CPU utilization

→ Time sharing systems
- switch the CPU among Processes frequently
- Users can interact with a Program while it's executing
- virtually run Processes at the Sam time

(ال Switch بيحصل بسرعة ف بيتهيألك إنهم شغالين Parallel وهم مش كده)
← زي إل فات ✓ ده اتشرح قبل كده :D)

Process scheduling Queues

ال Queue بصفة عامة ← First in First out
← غالباً ال OS مش بيتعامل بالمبدأ ده ، بيكون عنده Algrothims تساعده يقرر انهي
اول واحدة هيشتغل عليها . تعالى نشوف أنواع ال Queues

① Job Queue
- set of all Process in the system

(خلينا نتفق إن كل Process بتتحط ف Queue معينة حسب حالتها)
∴ هنا هيكون فيه ال new Processes في ال Job Queue

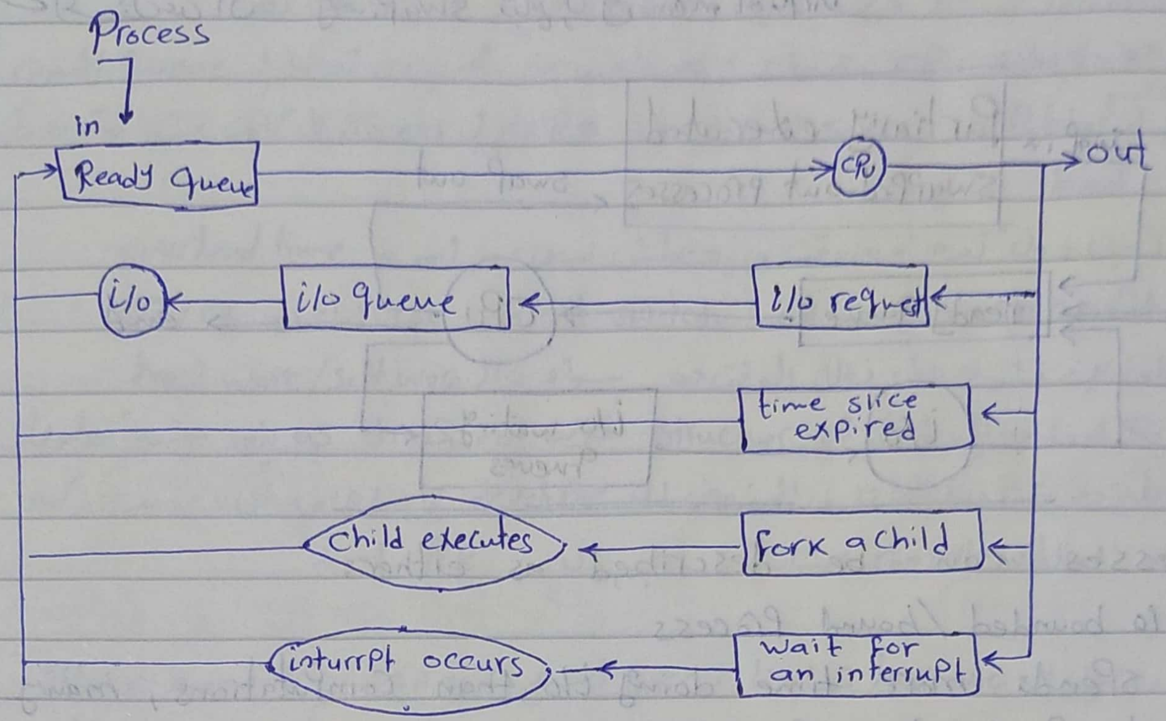
② Ready Queue
- Set of all Processes residing in main memory which is ready and waiting to execute.

③ Device Queue (waiting Queue)
- Set of Processes waiting for an i/o device

(*) Process migration between the various queues
← ع حسب ال status تباعتها ولكن بتبقى ف ال Job queue مرة واحدة فقط .

Ready Queue and various i/o Device Queues
ال head بيشاور ع اول PCB وكل PCB بيشاور ع ال بعده (Pointers)
وال tail بيشاور ع آخر PCB .. (4 Page 15)
وال OS هو ال بيقرر مين يدخل ال CPU اسمه PCB scheduler

Represention of Process scheduling



Schedulers

مجموعة من ال Algrothims ال بتختار انهي Process هياخدها من ال queue وتروح لل CPU

وله ٣ أنواع

① long-Term scheduler or job scheduler

↳ select whic Processes should be into the ready queue.

تحديد انهي Process من ال new تروح وتبقى ready فهو بيشتغل لما ال Process يحصلها creation

طب ما هو الطبيعي ال new تبقى ready أصلا ... قالك لأ ده: لأن ال Process أول ما يحصلها creation المفروض loading في ال main memory

طب افرض مفيش مساحة بقا في ال long-term-sch هو بيشوف لو فيه مساحة لل Process يدخلها في ال ready queue

وبالتالي هو بيقدر يحدد عدد ال processes اللي حصلها loading في ال main memory وبالتالي يقدر يعرف ال degree لل multi Programming

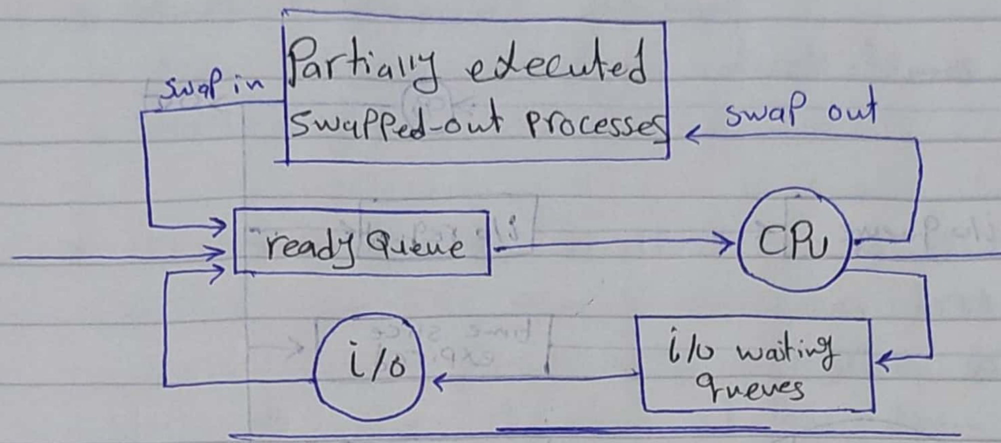
↳ executes infrequently, may be absent in some o/s

② short term scheduler or CPU scheduler

Select which Process should be executed next and allocates CPU, executes frequently.

Addition of medium Term scheduling

المسؤل عن ال swaping تباع ال virtual memory



- Processes can be described as either:

① i/o bounded / bound Process
↳ spends more time doing i/o than computations, many short CPU bursts.

هي دي process بيكون مثلاً جزء من وقتها بتقضيه في i/o operations والباقي cpu operation وده لإنها مش بتحتاج ال cpu لوقت كبير لكن بتحتاج أكتر i/o

② CPU bound Process
↳ spends more time doing computations, few very long CPU bursts. (ودي بقى عكس ال فاتت :D)

طب ال os الكويس بيشتغل بأنهي فيهم (يعني mix بين الاتنين)

- Proper system performance
↳ mix of CPU & i/o bound Processes
- improper system performance
↳ All Processes are i/o bound

يعني كده كلهم في ال ready queue وبالتالي ال os مش قادر يختار مين يدخل وبالتالي ال cpu هتكون idle

↳ All Processes are cpu bound

هيبقى كلهم في ال ready queue بيتصارعوا ع ال cpu ومفيش ولا واحدة بتعمل i/o operation

يبقى الأفضل ال os يشتغل ب mix من الاتنين وهيعرف يفصل ده عن طريق ال PCB بتاع كل Process

Context switch

. لو لأي سبب ال os قرر يشيل Process ويحط واحده مكانها

① ال CPU scheduler ← هيقرر أنهي واحده من الموجودين تدخل ال ready queue

← يجي ال dispatcher (s.w) يـ save المعلومات في ال PCB تبع ال Process اللي كانت شغاله وبعدكده يـ load التانيه

في الاتنين دول كده فيه وقت وبيحصل فيهم اسمه overhead time

← ال overhead time ده هو ال context switch عشان أقلل ①، ②

طب عاوزين نشوف حل نقلل الوقت ده سواء بال hardware/software

لما اتكلمنا عن ال structure of PCB قولنا إن ممكن ال os يزود شوية معلومات تانيه فده كده هيزود وقت نقل الداتا في المفروض ال developer يراعوا حاجه زي دي عشان كده هيزود ال context switch عشان كده حجم ال PCB زاد.

← فيه حاجات hardware تاعنا نشتغل أسرع فيه بروسيسور اسمه SPARCK زود عدد ال registers ← (بيعمل save & load لعدد ال registers يقفلوا off بعض ال Process تفضل طالعه وطارحه من ال register وهو مش هيسيف (ال os بعرف) ال لو ال registers دول اتملوا في ساعتها هيضطر يـ save عشان يـ switch ويـ load واحدة تانيه (Process) في كده بيعمل switch أقل.

- When CPU switches to another Process, the system must save the state of the old Process and load the saved state for the new Process.

- Context-switch time is overhead; the system doesn't do useful work while switching.

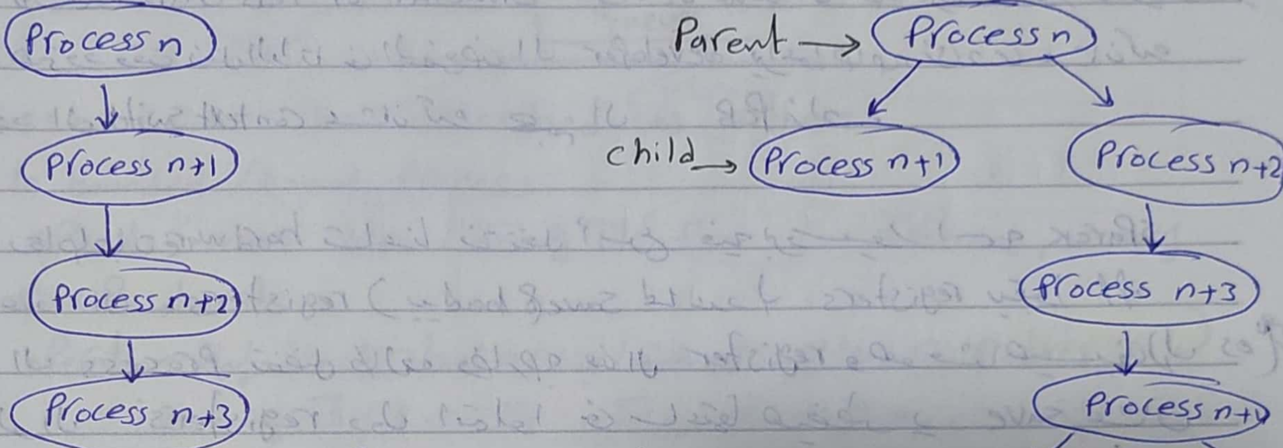
- Time dependent on hardware support.

Operations on Processes

قولنا قبل كده أن ال Process بيحصلها Process creation & Termination
وقولنا ال resources بتاعة ال Process (cpu time, memory, files, i/o devices)
تعالى نمسك واحده واحده ونشرحها أكتر

Process creation

الأول كده ال Process ينفع من خلالها نعمل creation لـ other Processes
ودة بيحصل بشكل من اتنين



• ال بيحدد الطريقة اللي هتشتغل بيها الـ ... هو الكود اللي كتبه ال developer لل Process
كمان ازاي ال resources هتتوزع ما بين ال Parent & child بتاعه

• Resource sharing
① Parent and children share all resources
or ② children share subset of Parent's resources
هنا ال Parent هيقسم ال resource ع ال children بتوعه بالتساوي.

• Execution
① Parent and children execute concurrently.
or ② Parent waits until children terminate.

• Address space
↳ child duplicate of Parent (هنا ال child نسخة طبق الأصل من ال Parent ولكن ب input مختلف)
or ↳ child has a program loaded into it

• unix examples
① fork (function) system call creats new process
② exec system call used after a fork to replace the process memory space with a new program
في ① هتعمل زي ال Parent
في ② ... ... exec بعد ال fork

Process Termination

- process executes last statement and asks the os to decide it (exit)
  - ↳ output data from child to Parent
  - ↳ Process' resources are de-allocated by os
- Parent may terminate execution of children Processes (abort)
  - ↳ child has exceeded allocated resources
  - or ↳ Task assigned to child is no longer required
  - or ↳ Parent is exiting
    - ↳ operating system doesn't allow child to continue if it's Parent terminates.
    - ↳ Cascading termination → ...

Cooperating Processes

- independent Process can't affect or be affected by the execution of another Process.
- cooperating Process can affect or be affected by the execution of another Process.

Advantages of Process cooperation

information sharing, Computation speedup, modularity, Convenience

inter-Process Communication

- message Passing
- shared memory
- synchronization

message Passing

message Passing → بتكون بين الـ Processes

الـ messages هتبقى نوعين
- Fixed size
- Variable size

الـ Fixed
- أسهل بالنسبة للـ OS
- أصعب بالنسبة للـ developer

الـ Variable
- أصعب للـ OS
- أسهل للـ developer

• Communication
- direct → send(P2, message), recive(P1, message)
- indirect → send(ID, message), recive(ID, message)

في الـ direct لو حصل تغيير في اسم الـ process الـ developer هيغير الكود كله (مش عملي)

في الـ indirect الـ developer ميعرفش الـ ID ولكن لما يـ create الـ child بيجي له (Pointer بيشاور ع الـ process فـ يقدر يعرف الـ ID بتاعها → الـ OS ليه معلومات الـ Processes لأنه المتحكم عليها وهو معينها)

Synchronization

• message passing may be either blocking or non-blocking

Send
- blocking → Process 1 هتبعت message لـ P2 وتفضل مستنية رد أو وصولها
- non blocking → Process 1 هتبعت message لـ Process 2 وتمشي

recive
- blocking → P1 هتفضل متعملش حاجة لحد ما تستقبل من P2
- non blocking → P1 هنا بيقول للـ OS أنا مش مستني رسالة بس لو جاتلي رسالة ضيف البت اعلى interrupt وقولي

Blocking is considered synchronous

nonBlocking is " Asynchronous

Blocking ex: TCP & UDP → non blocking

هنسترجع شوية Concepts حولناهم قبل كده

- maximum CPU utilization obtained with multitasking

كل ما كان ال idle time أقل كل ما كان أحسن بالنسبة لل OS فده بيحصل عن طريق ال multitasking فال CPU مش قاعدة فاضية وشغالة.

- CPU - i/o Burst cycle

↳ Process execution consists of a cycle of CPU execution and i/o wait.

→ ف الوقت بتاع ال Burst cycle وقت بيقضيه مع ال CPU ووقت مع ال i/o
(+ Page 6 ف السلايدز بتوضح الشكل ده)

CPU Scheduler

- Selects from among the processes in memory that are ready to run, and allocates the CPU to one of them.
- CPU scheduling decisions may take place when a process
  1. switches from running to waiting state
  (waiting for i/o or waiting for interrupt or child finish execution)
  2- switches from running to ready state
  ↳ at interrupt or finished it's timeslot
  3- switches from waiting to ready
  4- Terminates ← لما ال process خلاص يخلص

CPU scheduler

فيه منه نوعين

① Preemptive

لو أنا شغال ع ال CPU وجه interrupt او time slote خلصت (زي process رجعت :D)
فأنا هيجيلي وقت أطلع من ال CPU وأروح ال Queue وأرجع لل CPU تاني ده اسمه Preemptive

- Process release the CPU befor it finish execution

ex: Modern OS : unix, Linux, windows 7

② non-Preemptive → بتطلع بس لما تخلص

↳ process release CPU when
  ↳ from Running to → waiting
  or ↳ from Running to → Terminted

ex:
→ microsoft windows 3.1

Dispatcher

بيـ Save المعلومات الـ Process اللي انقطعت ما cpu وبـ load الجديدة غيرها

Gives control of the cPU to the Process selected by the short-term schedulers:-

① switching context (save & load)

② switching to suitable mode (user or monitor)
(user if the process is user process
monitor mode if " " " system Process)

③ jumping to the proper location in the user program to restart that program (program counter اللي فيه pc) register بيسجل فيه عنوان الـ inedx

- Dispatch latency
  ↳ = overhead time هو ده
  ↳ time taken by dispatcher to stop one process and start another running.

Scheduling criteria

من خلالها بحكم ع الـ Algrothim اذا كانت كويسة او لا

① cPu utilization → keep the cPu as busy as possible.

② ThroughPut → num of processes that complete their execution per time unit

③ Turnaround time → amount of time to execute a particular process (الوقت من اول ما create الـ Process لحد ما اتعملها Terminted (وقت تنفيذ الـ Process كله))

④ waiting time → amount of time a process has been waiting in the ready queue (مش الـ waiting queue) لحد ما ادخل ع الـ cPu

⑤ Response time → amount of time it takes from when a request was submitted until the first response is produced, not output
↪ (for Time-sharing environment)

الوقت الـ هياخده الـ Process من اول ما يعملها creation لحد اول مرة تدخل الـ cPu
(new → running)

optimization criteria

(حاجات قسمنا وخليناها احسن)

- maximize (CPU utilization , ThrougPut)
- minimize (Turnaround time, waiting time, Response time)
- considerations
  - minimize maximum response time
  - minimize the variance of response time

Scheduling Algorithms

- first-come first served
- shortest-job first
- Priority
- Round-Robin

FCFS Scheduling

- easily implemented
- Ready queue is FIFO → first in first out
- P_n ready → P_n PCB is linked to tail of queue
  هنا يقصد ان آخر واحدة بتيجي هي بتكون آخر واحدة في ال queue
- Process at head of ready queue → CPU
  اول واحده جت بقا هي اللي هتدخل ع ال CPU
- average waiting time is long

ex:

Process	Burst time
P_1	24
P_2	3
P_3	3

Suppose that the processes arrive in the order P_1, P_2, P_3
The Gantt chart for the schedule is

P_1	P_2	P_3
0 – 24	24 – 27	27 – 30

(في الحالة دي بس)

waiting time = responce time for $P_1 = 0$, $P_2 = 24$, $P_3 = 27$
Average waiting time = $(0 + 24 + 27)/3 = 14$

ex2:

Process	Burst time
P_1	3
P_2	3
P_3	24

P_1	P_2	P_3
0 – 3	3 – 6	6 – 30

waiting time = responce for $P_1 = 0$, $P_2 = 3$, $P_3 = 6$
Average waiting time = $(0 + 3 + 6)/3 = 3$

SJF

Associate with each process the length of its next CPU burst. use these lengths to schedule the process with the shortest time

- Two schemes:

① Non Preemptive → once CPU given to the process it cannot be Preempted until Compleletes its CPU burst.

② Preemptive → if a new process arrives with CPU burst length less than remaining time of current executing process, Preempt. (SRTF)

↳ This scheme is know as the shortest Remaining Time first

- SJF is optimal

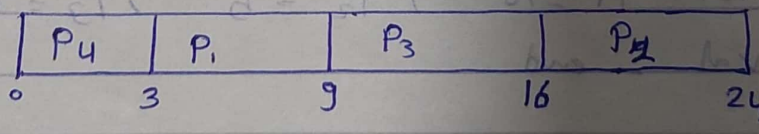
↳ gives minimum average waiting time for a given set of processes

أقل وقت في الانتظار ← Algorithms

ex:

Process	Burst time
P_1	6
P_2	8
P_3	7
P_4	3

ملحوظة: هنا الـ Processes كلهم وصلوا في وقت واحد.

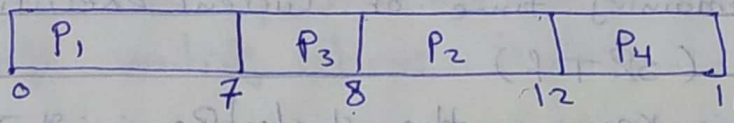


waiting time for $P_1 = 3$, $P_2 = 16$, $P_3 = 9$, $P_4 = 0$

average time = $(3 + 16 + 9 + 0)/4 = 7$

ex of non-Preemptive SJF

Process	Arrival time	Burst time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



average waiting = (0 + 6 + 3 + 7) / 4 = 4

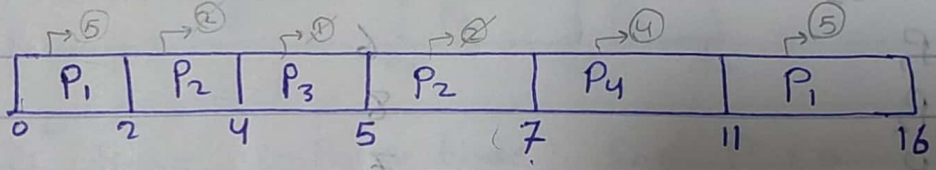
8-2 , 7-4 , 12-5

waiting = start - arrival

Turnaround $P_1 = 7$, $P_2 = 10$, $P_3 = 4$, $P_4 = 11 = (16-5)$

ex: of Preemptive SJF

Preemptive



waiting time for $P_1 = 9 = (11-2)$, $P_2 = 1 = (5-4)$

$P_3 = 0$ $P_4 = 2 = (7-5)$

Turnaround for $P_1 = 16$, $P_2 = 5$, $P_3 = 1$, $P_4 = 6$

↳ arrival - end

responce time for $P_1 = 0$, $P_2 = 0$, $P_3 = 0$, $P_4 = 2$ (7-5)

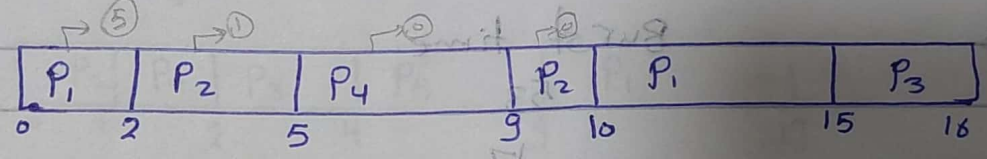
Average waiting = (9 + 1 + 0 + 2) / 4 = 3

Priority scheduling

- Priority number is associated with each Process
- The CPU is allocated to the Process with the highest Priority → (smallest integer ≡ highest Priority)
  - Preemptive
  - Non Preemptive
- SJF is a priority scheduling where Priority is the Predicted next CPU burst time.
- Problem → Starvation → low Priority Processes may never execute
- Solution → Aging → as time progresses increase the Priority of Process.

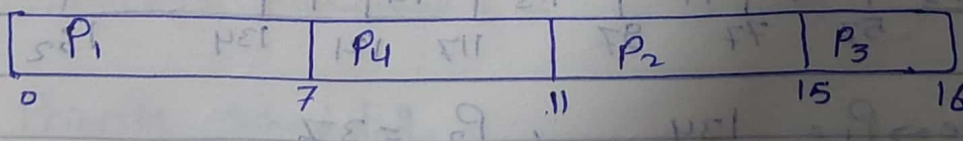
ex. of Preemptive Priority

Process	arrival time	Burst time	Priority
P_1	0.0	7	3
P_2	2.0	4	2
P_3	4.0	1	4
P_4	5.0	4	1



Average waiting time = (8 + 4 + 11 + 0)/4 = 23/4

for non preemptive Priority



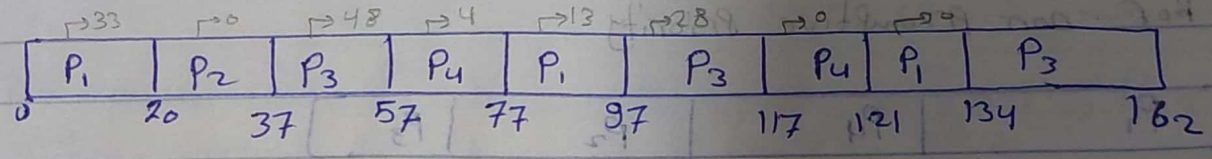
Average waiting time (0 + 9 + 11 + 5)/4 = 25/4

Round Robin (RR)

- Each process get a small unit of CPU time (time quantum), usually 10-100 ms. after this time has elapsed, the process is preempted and add to the end of the ready queue.
- if there are n processes in the ready queue and the time quantum is q, then each process gets 1/n of the CPU time in chunks of at most q time units at once. No process waits more than (n-1)q time units.
- Preformance
  - If q large ⟶ FIFO
  - q small q must be large with respect to context switch, otherwise overhead is too high.

ex of RR, Time Quantum = 20

Process	Burst time
P_1	53
P_2	17
P_3	68
P_4	24

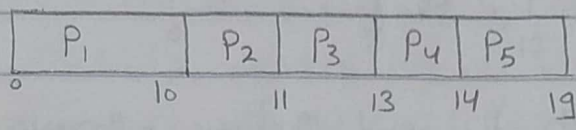


turnaround time ⇒ P_1 = 134, P_2 = 37
P_3 = 162, P_4 = 121

responce time P_1 = 0 P_2 = 20
P_3 = 37 P_4 = 57

Process	Burst time	Priority
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

FIFO



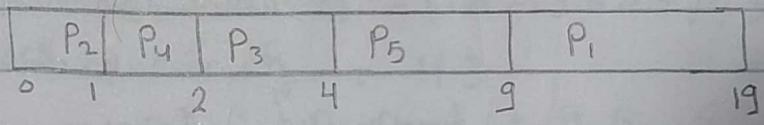
waiting time $P_1 = 0$, $P_2 = 10$, $P_3 = 11$, $P_4 = 13$, $P_5 = 14$

average waiting = $(0+10+11+13+14)/5 = 9.6$

average execution = $(10+1+2+1+5)/5 = 3.8$

average turnaround = $9.6 + 3.8 = 13.4$

sjf non Preemptive



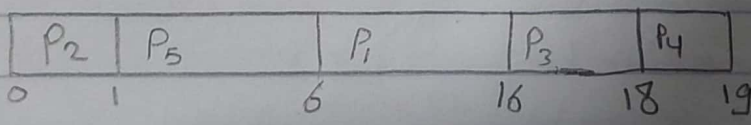
waiting time $P_1 = 9$, $P_2 = 0$, $P_3 = 2$, $P_4 = 1$, $P_5 = 4$

average waiting = $(9+0+2+1+4)/5 = 3.2$

average execution = 3.8

average turnaround = 7

Priority scheduling

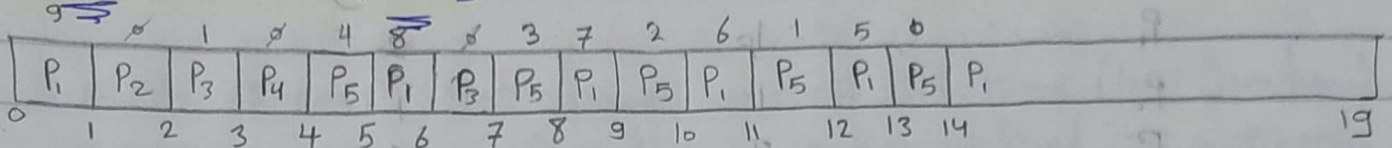


average waiting time = $(6+0+16+18+1)/5 = 8.2$

average execution = 3.8

average turnaround = 12

RR Q = 1



waiting time

$P_1 = (0+4+2+1+1+1) = 9$

$P_2 = 1$

$P_3 = (2+3) = 5$

$P_4 = 3$

$P_5 = (4+2+1+1+1) = 9$

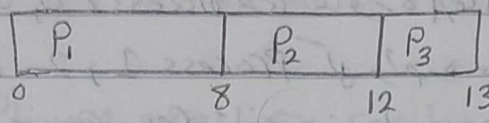
average waiting $= (9+1+5+3+9)/5 = 5.4$

average execution $= 3.8$

average turn around time $= 5.8+3.8 = 9.2$

Process	arrival time	Burst time
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

FCFS

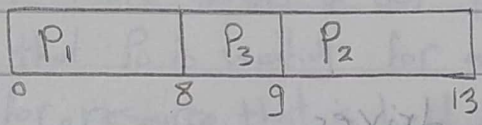


average waiting time = (0 + 7.6 + 11) / 3 = 6.2

average executing = (8 + 4 + 1) / 3 = 4.3

average turnaround = 6.2 + 4.3 = 10.5

SJF non Preemptive



average waiting = (0 + 8.6 + 7) / 3 = 5.2

average executing = 4.3

average turnaround = 9.5

operating system fundamentals

Session 5 [Deadlock]

الـ Deadlock معناه قفله حصلت ف الجهاز بمعنى إن فيه مجموعة من
الـ Processes بتتخانق ع الـ resources وحصلت بينهم قفله.
ف أنت بيكون عندك Process 1 معاها resource معين وليكن A وفنفس الوقت
بتطلب resource B مثلاً. ف هيروح الـ os لـ resource B هيلاقي حاجه
وفي Process 2 الـ Process 2 دي عاوزه resource A
ف كده عندك Process 1, 2 كل واحده معاها resource معين (وفي نفس
الوقت عاوزه resource تانى عند الـ process التانيه ومفيش واحده فيهم
هتسيب الـ resource للتانيه والـ resource نفسه مش sharable عشان
تشتغل عليه التانيه ف الوقت ده أو sharable بس لـ instance معينه.

A set of blocked processes each holding a resource and waiting to acquire a resource held by another process in the set

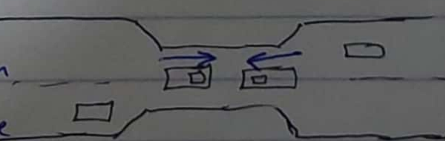
Example

- system has 2 tape drives
- P_1 and P_2 each hold one tape drive and each needs another one.
- semaphores A and B, initialized to 1

← الـ os بيستخدم حاجه اسمها semaphores ودى حاجه عامله زى الـ flag
عشان أقول ينفع الـ Process تعمل الحاجه دى دلوقتى ولا لأ
← لو ينفع يعنى Sem=1 / مينفعش يعنى 0

P_1	P_2	
wait (A)	wait (B)	← الـ Process معاها ده
wait (B)	wait (A)	← بس عاوزه ده برضو

Bridge crossing example



- → Traffic only in one direction
- → each section of a bridge can be viewed as a resource.
- → if a deadlock occurs, it can be resolved if one car backs up (Preempt resources and rolled back)
- → Starvation is Possible → شرحناه في المحاضرات
  كل ده يقا نقدر نطبقه ع الـ process

Deadlock characterization

عشان اقول حصل عندي deadlock لازم يتحقق أربع شروط (كلهم مع بعض)

① mutual exclusion : only one Process at time can use a resource

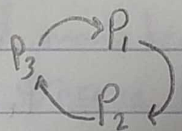
كل ما عدد ال instance زادت كل ما احتمالية ال deadlock قلت

② Hold and wait : a Process holding at least one resource is waiting to acquire additional resources held by other processes.

③ No Preemption : a resource can be released only voluntarily by the Process holding it, after the Process has completed its task

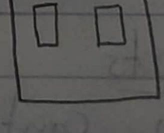
يعني ال OS مش هيتدخل ويعمل release لل resource دي من process ويديها لحد تاني في حالة ال no Preemption لكن لو فيه Preemption هيتدخل.

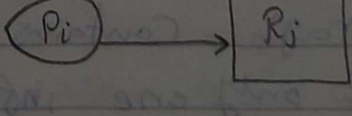
④ Circular wait : there exists a set $\{P_0, P_1, \ldots, P_n\}$ of waiting processes such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by P_2, …, P_{n-1} is waiting for a resource that is held by P_n, and P_n is waiting for a resource that is held by P_0

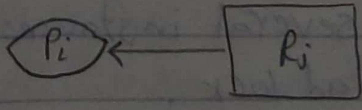


Resource - Allocation Graph

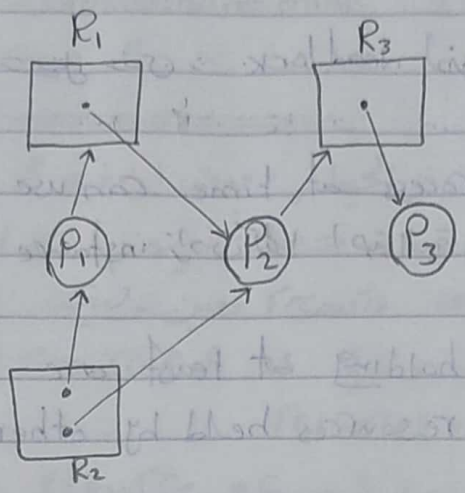
- Process 

- Resource type with 2 instances 

- P_i request instance of R_j 

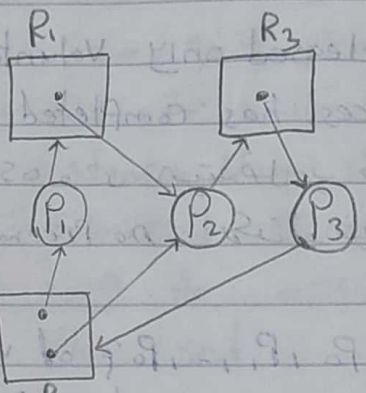
- P_i is holding an instance of R_j 

ex:



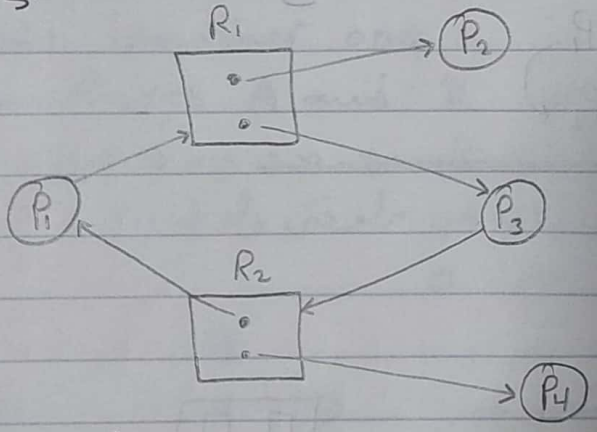
هنا ما فيش deadlock
لأن P3 هتستنى R3 وتحرر P2
و P2 هتستنى R1 وتحرر وكانوا P1

ex:



هنا فيه deadlock
لأن كل واحد منهم هتستنى R
عشان تاخد R تانية

ex:



هنا فيش deadlock
عشان P2 لو خلصت هتحرر P1
و P3 تحرر كمان P1

أو حتى P4 خلصت هيبقى فيش
الكلام

Basic facts

- if graph contains no cycles no deadlock.
- if graph contains a cycle
  - ↳ if only one instance per resource type, then deadlock
  - ↳ if several instances per resource type, possibility of deadlock.

Methods for Handling Deadlocks

هنعرف ازاى ال os بيهندل و نتعامل مع ال deadlock وده بيكونا
بناء على ال structor تاعه والمستخدم ال Algrothim الخاص به
لعدنه نتعامل مع ال deadlock اصلا

① Ensure that the system will never enter a deadlock state
← اناال os يعمل مجموعة من ال actions لا هيتأكد بيها ان مفيش اى deadlock هيحصل

② allow the system to enter a deadlock state and then recover
← يعنى ممكن ال os ميعرفش انه هيتأثر بال deadlock وممكن لما يتأثر بيه هيعرف يعمل recover منه

③ ignore the Problem and prented that deadlocks never occur in the system ; (used by most operating systems) ex: unix

Dead lock Prevention

① mutual Exclusion → not required for sharable resources ; must hold for non sharable resource (دمسمكن يكون نوع HW بيعمله أو SW زى ال API)

② Hold and wait

- must guarantee that whenever a process requests a resource, it doesn't hold any other resources.

↳ Require process to request and be allocated all its resources befor it begins execution, or allow process to request resource only when the resource has none. (يعنى لما ال P1 عاوزه R1 و R2 معا ال os هيديهملها اصلا قبل ما يطلب R2)

↳ low resource utilization, starvation Possible (انها ال P تفضل في ال wait queue لفترة طويلة)

③ No preemption (ال os بيسجل كل P محتاجه ايه وماسكه ايه عشان لما R الحاوزة يفضى يتولها)

- if a process that is holding some resources requests another resource that can't be immediately allocated to it, then all resources currently being held are released.
- Preempted resources are added to the list of resource for which the process is waiting
- Process will be started only when it can regain its old resources, as well as the new ones that it is requesting

④ circular wait (بيدى لكل R رقم ويمنع اى P تطلب R ارقام اقل من اللى هو فيها)

يعنى لو P1 ماسكه R3 ينفع تطلب R4, ... لكن متطلبش R1, R2

impose a total ordering of all resource types, and require that each process requests resource in an increasing order of enumeration.

Recovery from deadlock : ~~Resource Preemption~~

① abort all deadlocked Processes (هموتهم كلهم يعني :D)

② abort one Process at a time until the deadlock cycle is eliminated

↳ بفضل بموت فيهم لحد ما ال deadlock ينتهي يعني لو هم 5 وهنتهي عند 3 هيقف
يعني يموت P_1 ويشوف هل ال deadlock انتهى لو لا هيموت P_2 ويشوف هل انتهى
لو اه مثلا خلاص مش هيموت تاني ويسيب الباقي شغال.

in which order should we choose to abort? (ال P نرتبهم ازاي ↓)

① - Priority of the Process.

② - How long process has computed, and how much longer to complete.
↳ يعني ال P اشتغلت اد ايه وفاضلها اد ايه وتخلص

③ - Resources the process has used

④ - Resources Process needs to complete

⑤ - How many Processes will need to be terminated

⑥ - is process interactive or batch?
↳ يعني فالأغلب ال P ال interactive هي اللي تبقى أهم عشان أموتها بعمل مع ال user
ولو شغالة لوحدها في ال background يعني batch

Recovery from deadlock : Resource Preemption

① selecting a victim → to minimize cost
أنا بختار victim process بناءً على ال 6 شروط ال فاتوا عشان أقلل ال cost.

② Roll back
↳ Return to some safe state, restart process for that state
(يعني انت كده فرضًا R معينة في P فتسحبها منه عشان ال R تبقى free تاني تعمل roll back وترجعها لل P)

③ Starvation
↳ Same process may always be picked as victim, include number of roll back in cost factor.