

Assignment

General Description

The following outlines the guidelines for a theoretical-practical activity for students in the Operating Systems course. The activity specifically assesses their mastery of some basic concepts, definitions related to CPU management, processes, scheduling, and the implementation of a simulator using C or C++.

Objectives During the development of the activities, students will achieve the following:

- Enumerate the milestones in the evolution of computer systems.
- Describe the objectives and functions of modern operating systems.
- Simulate the behavior of the CPU and process management.
- Apply the principles of scheduling.

Before You Begin

Read the following:

- *Operating System Concepts* (9th Edition) by Silberschatz and Galvin: Chapters 1, 2, and 3
- *Modern Operating Systems* by Tanenbaum: Chapters 1 and 2

Create a git repository and add the link:

Link: <https://github.com/Juan-Ospina1216/Lab-1-SO>

Integrantes: Sebastian Enriquez, Juan Pablo Ospina y Jhon Silva

Activity No. 1: Conceptualization [10%]

After completing the recommended readings in this document, answer the following questions in your own words, concisely and accurately.

1. Enumerate the milestones in the evolution of computer systems.

R/ En las cuatro generaciones de la evolución de los sistemas de cómputo existen varios hitos que son:

- Primera generación: Los bombillos al vacío.
- Segunda generación: Los transistores.
- Tercera generación: Circuitos Integrados.

- Cuarta generación: Aparición de los Pc's.
- 2. What are the four components of a computer system? Describe each one.
 - Hardware: Se refiere a la CPU, la memoria y los dispositivos de entradas y salidas.
 - Aplicación de programas: Define la forma en que se usan estos recursos para resolver los problemas de cómputo de los usuarios.

- Sistema Operativo: Se conoce como una extensión de la maquina donde se pueden ejecutar muchas tareas.
 - Usuarios: Introducen datos e interactúan con el sistema de cómputo a través de programas de aplicación.
3. What is the difference between a monolithic kernel and a microkernel?
 - Monolithic: Aquí todo el sistema operativo se ejecuta en un solo espacio de direcciones, especialmente en modo Kernel. Cuando se ejecuta pone todo en memoria por si en la aplicación se necesita algo.
 - MicroKernel: Aquí solo las funciones esenciales, como los procesos, la gestión de memoria se ejecutan en el modo kernel. Otras funciones se ejecutan en modo usuario. Por ejemplo, si una aplicación necesita algo esencial se carga en modo kernel, pero de manera perezosa.
 4. Define an Operating System from two different perspectives.
 - Top Down: Una version extendida de la maquina física.
 - Bottom Up: El SO administra recursos como la CPU, la memoria y el disco.
 5. What is the purpose of system calls?
 - El llamado al Sistema le permite al usuario decirle al SO qué hacer. Especialmente sirven para correr programas, acceder a memoria, y acceder a dispositivos.
 6. What is a process?
 - Un proceso se puede definir de varias formas. Se le conoce como una entidad activa en ejecución, un programa en ejecución, y también como una instancia de un programa.
 7. What is a multiprogrammed operating system?
 - Permite la ejecución simultánea de múltiples programas en una computadora, logrando aprovechar el tiempo de inactividad de la CPU durante la espera de una solicitud de I/O. Esto conlleva a maximizar el uso de la CPU y mejorar la eficiencia general del sistema.
 8. What are the states of a process?

R/ Existen como mínimo 3 estados y máximo 7. El de 3 estados se compone de:

 - Estado de ejecución.
 - Estado de Listo.
 - Estado de bloqueado.
 9. What information is stored in the Process Control Block (PCB) associated with a process?
 - La PCB sirve como depósito de cualquier información que pueda variar de un proceso a otro. El PCB contiene:
 - o Estado del proceso, el contador de programa, registros de la CPU, información de la memoria, información del estado de las I/O.
 10. What are the main activities of an operating system in relation to process management?
 - Programación de procesos e hilos en las CPUs.
 - Creación y eliminación de procesos de usuario y sistema.
 - Suspende y reanuda procesos.
 - Mecanismos de sincronización de procesos.
 - Mecanismos para la comunicación de procesos.

Activity No. 2: Remembering the C and C++ languages [20%]

1. Implements the programs and functions below:

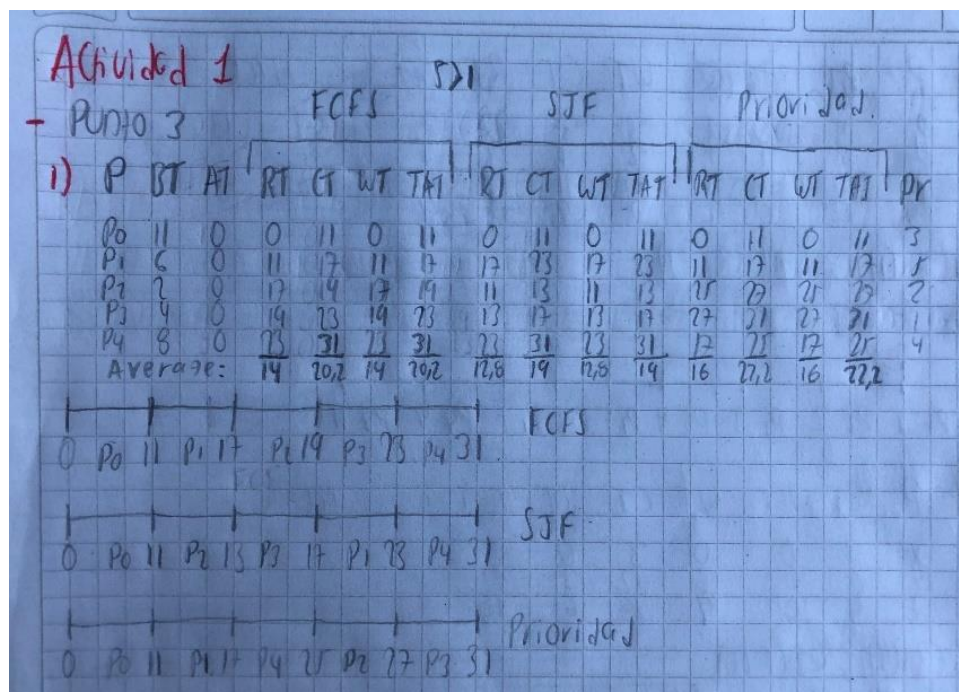
Write a program in c to find the leap year.
Write a program to calculate factorial of a number.
Write a program in c to calculate power using recursion.
Write a program in c to find even or odd numbers.
Write a program in c to print fabonnaci series.
Write a Function to check uppercase letter.
Write a program in c to function to check lowercase letter
Find the greater of the three numbers
Write a program in c to type casting implicit explicit.
Write program to display number 1 to 10 in octal, decimal and hexadecimal system.

2. Write a C program that receives and processes grades for the Operating Systems (SO) course using structures.
3. Write a C++ program that performs the following tasks: First, the program should receive a numeric value and determine whether it is a prime number, displaying the result. Second, the program should accept a list of numbers and identify the prime numbers within that list. Lastly, the program should allow the user to input a numeric range and display all prime numbers within that specified range.
4. Create a program with C++ to model geometric shapes and perform area, perimeter, and color operations. Rectangle, square, triangle, etc. shapes must be considered. Tip. Use classes and inheritance.

Activity No. 3: CPU Scheduling [20%]

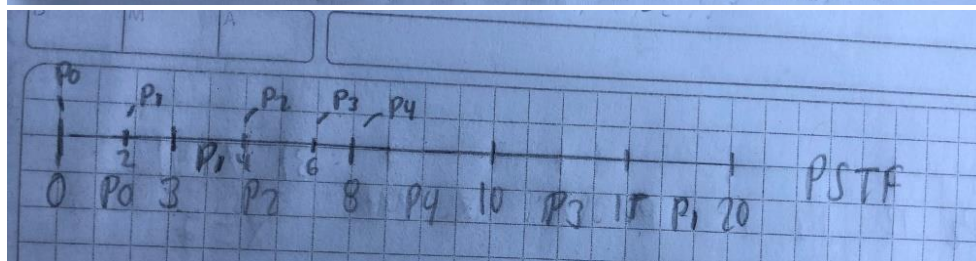
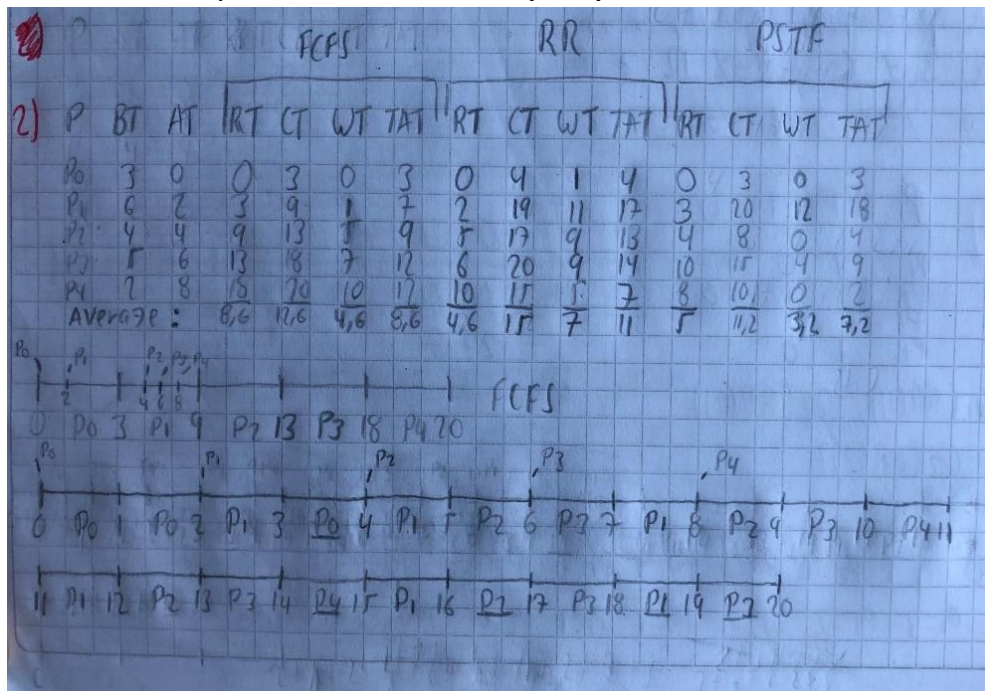
1. Five processes arrive at the ready queue simultaneously with estimated execution times of 11, 6, 2, 4, and 8 units of time, and priorities of 3, 5, 2, 1, and 4 respectively, where 5 represents the highest priority. Present the execution, metrics, and Gantt chart for the following scheduling policies: First-Come, First-Served (FCFS): Order of execution 11, 6, 2, 4, 8, Shortest Job First (SJF) and Priority Scheduling.

R/ El primer diagrama de Gantt representa la política FCFS, el segundo es Shortest Job First y el último es el de prioridad.



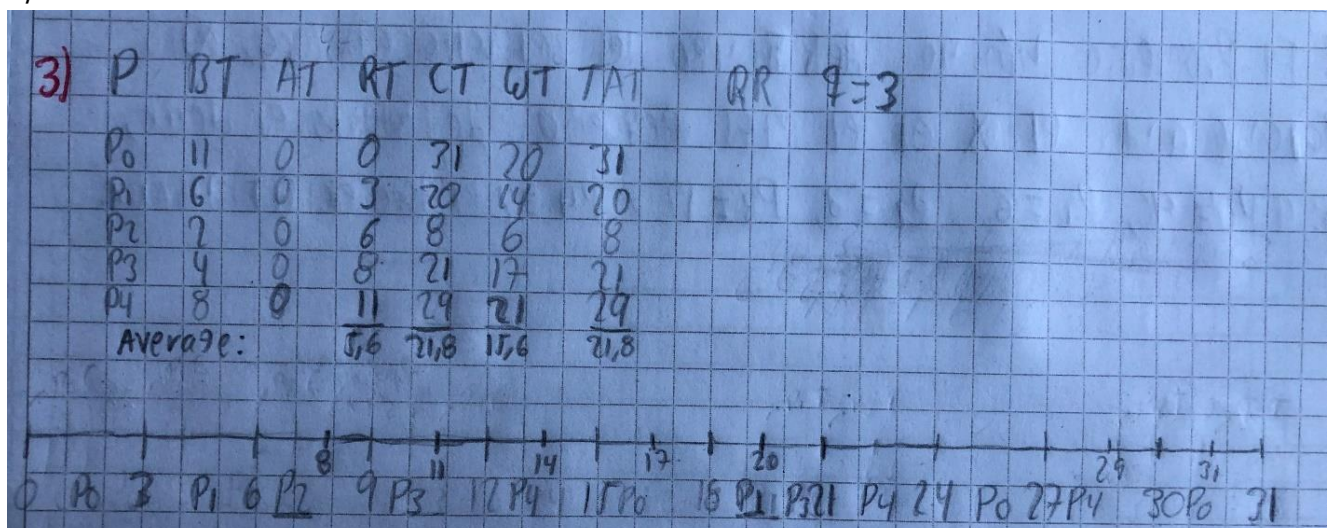
2. Five processes arrive at the ready queue at times 0, 2, 4, 6, and 8, with estimated execution times of 3, 6, 4, 5, and 2 units of time. Present the execution, metrics, and Gantt chart for the following scheduling policies: First-Come, First-Served (FCFS): Order of execution 0, 2, 4, 6, 8. Round Robin (RR) with a quantum of 1 unit and Preemptive Shortest Job First (SJF).

R/ El primer diagrama de Gantt representa la política FCFS, el segundo es Round Robin y el último es Preemptive Shortest Job First (PSJF).



3. Five processes are waiting for execution with estimated execution times of 11, 6, 2, 4, and 8 units of time, using the Round Robin (RR) scheduling policy with a quantum of 3 units. Present the execution, metrics, and Gantt chart.

R/

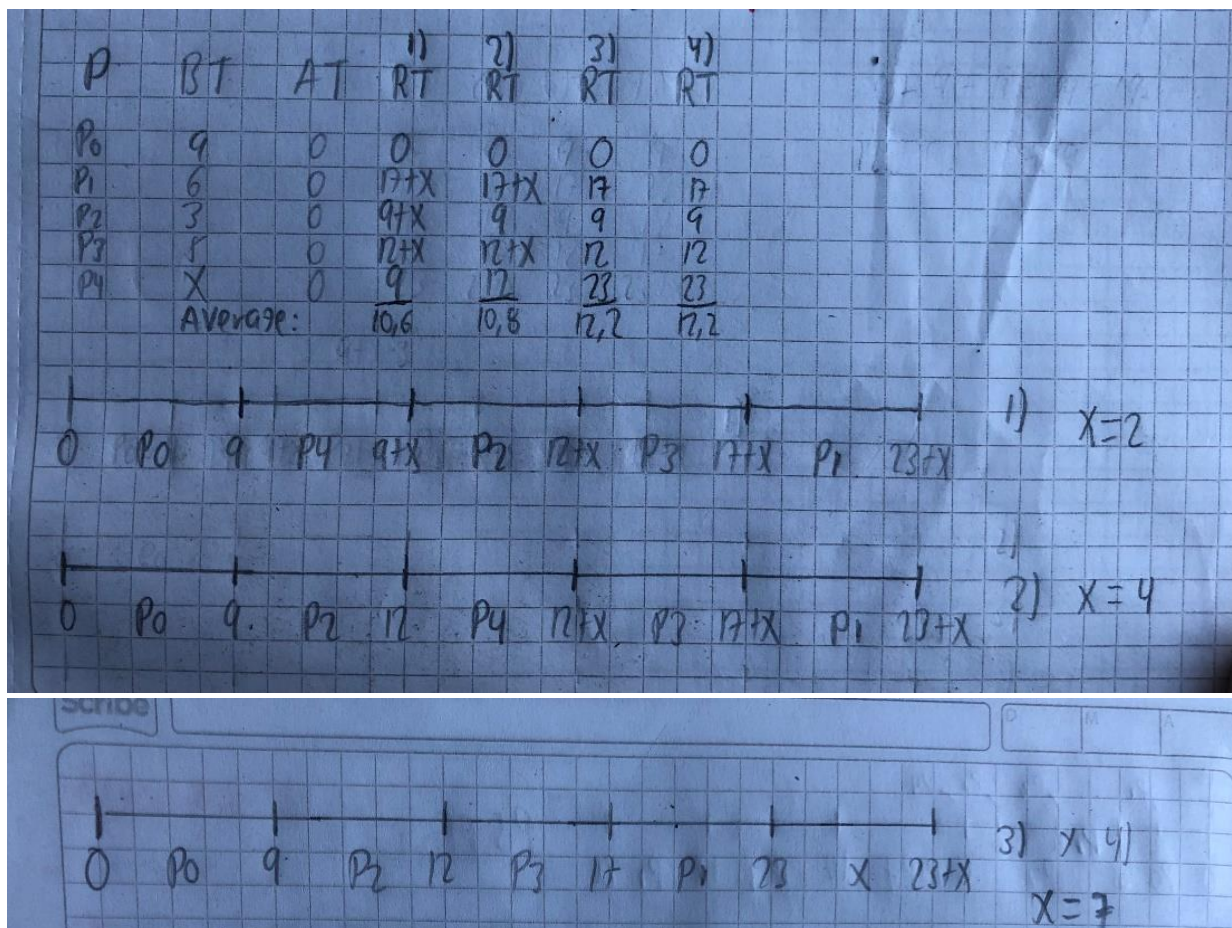


4. Five processes are waiting for execution with estimated execution times of 9, 6, 3, 5, and X units of time. Determine the order of execution that minimizes the average response time, noting that the answer depends on the value of X.

R/ Para este punto se usará la política SJF, ya que es aquella que atiende primero a los procesos con menor tiempo de ejecución, por lo que en este caso sería beneficioso puesto a que el tiempo de llegada de los procesos son iguales. Sabiendo que la respuesta depende de X se va a asumir que:

1. $X < 3$
2. $3 \leq X \leq 5$
3. $6 \leq X \leq 9$
4. $X > 9$

Cada tiempo de respuesta y diagrama de Gantt le va a corresponder un número del 1 – 4 para mostrar cada una de las situaciones. El primer diagrama muestra la primera situación, la segunda muestra la segunda situación y el último muestra tanto la 3 como la cuarta condición. Cabe recalcar que el último diagrama de Gantt es el mismo para la suposición 3 y 4, ya que, aunque se escoja cualquier valor de acuerdo con su rango, siempre va a ser el último en procesar.



Al final se evidencio que el promedio mínimo de tiempo de respuesta es 10.6, específicamente cuando X es el mas pequeño de todos, y se pudo notar justamente cuando el X se evaluó con respecto a cada una de las condiciones.

Activity No. 4: Instruction Cycle Simulator in C [40%]

Create a program in C++ that simulates the basic instruction cycle. The simulator should support the following instruction set:

SET - MEM - *SET D1 X NULL NULL*, Store X value in D1 memory address. where X is an immediate, direct or constant value. When SET instruction is read the X value is stored in Memory without processor execution.

LDR - LOAD - *LDR D3 NULL NULL* Load the value in D3 memory address and puts in accumulator register

ADD - ADDITION - There are three ways: *ADD D1 NULL NULL*, adds the value in D1 memory address to loaded value in accumulator register. *ADD D1 D3 NULL* Load the value in D1 memory address in the accumulator register and add to found value in D3 memory address. *ADD D1 D3 D4* same that *ADD D1 D3* but puts the result in D4

INC - INCREMENT - *INC D3 NULL NULL* Load the value in D3 memory address adds 1 and store in same address

DEC - DECREMENT - *DEC D3 NULL NULL* Load the value in D3 memory address adds 1 and store in same address

STR - STORE - *STR D3 NULL NULL* Read the value in accumulator register and puts in D3 memory address

SHW - SHOW - *SHW D2 NULL NULL* show the value in D2 memory address, *SHW ACC* show the value in accumulator register, *SHW ICR* show the value in ICR register, *SHW MAR* show the value in MAR register, *SHW MDR* show the value in MDR register, *SHW UC* show the value in Control Unit.

PAUSE - PAUSE - *PAUSE NULL NULL NULL* stop the instruction cycle and allow SHOW register values and memory values at any time

END - FINISH READING INSTRUCTION

The simulator must read plain text files with executable instructions. For example, below is the content of a plain text file named *programa1.txt*.

```
SET D5 12 NULL NULL
SET D2 23 NULL NULL
SET D8 3 NULL NULL
SET D3 5 NULL NULL
LDR D2 NULL NULL NULL
ADD D5 NULL NULL NULL
ADD D8 NULL NULL NULL
STR D3 NULL NULL NULL
LDR D3 NULL NULL NULL
ADD D2 NULL NULL NULL
STR D2 NULL NULL NULL
SHW D2 NULL NULL NULL
END NULL NULL NULL
```

and show the result

61

1. Create three 'programs' (plain text files) that the simulator can read, load into 'memory', fetch, decode, execute, and present the results.

To consider:

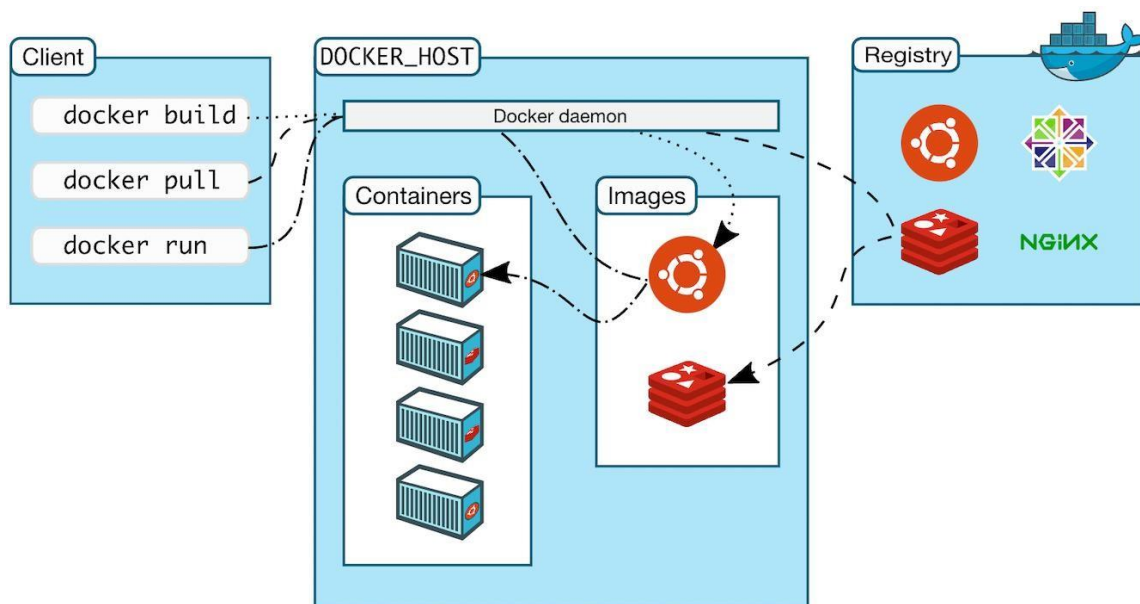
- ** Only one program is executed at a time.
- ** Implement the main memory as an array.
- ** Implement each CPU register independently."

Activity No. 5: Docker practical session [10%]

Docker is a command line-based software allowing users to manipulate images and create application containers.

As presented in the course, Docker consists of two elements:

- a client, to receive commands from the user
- a server, to execute commands and manage images and containers



Docker commands architecture

Typing this command will give the Client and Server versions available on your computer.

Paste a screenshot of result

```
docker version
```

```
C:\Users\sebat>docker version
Client:
 Version:           27.0.3
 API version:       1.46
 Go version:        go1.21.11
 Git commit:        7d4bcd8
 Built:             Sat Jun 29 00:03:32 2024
 OS/Arch:           windows/amd64
 Context:           desktop-linux
```

Usage, options and a full list of available commands can be accessed through the command line in a terminal. Type the following command

```
docker --help
```

```
C:\Users\sebat>docker --help
```

```
Usage:  docker [OPTIONS] COMMAND
```

```
A self-sufficient runtime for containers
```

```
Common Commands:
```

run	Create and run a new container from an image
exec	Execute a command in a running container
ps	List containers
build	Build an image from a Dockerfile
pull	Download an image from a registry
push	Upload an image to a registry
images	List images
login	Log in to a registry
logout	Log out from a registry
search	Search Docker Hub for images
version	Show the Docker version information
info	Display system-wide information

```
Management Commands:
```

builder	Manage builds
buildx*	Docker Buildx
checkpoint	Manage checkpoints
compose*	Docker Compose
container	Manage containers
context	Manage contexts
debug*	Get a shell into any image or container
desktop*	Docker Desktop commands (Alpha)

The general usage of a Docker command line is as follows:

```
docker [OPTIONS] COMMAND [arg...]
```

Questions

1. How many arguments are absolutely required by the command 'docker pull' ?

R/ Se necesita exactamente 1 argumento

```
What's next:  
View a summary of image vulnerabilities and recommendations → docker scout quickview  
"docker pull" requires exactly 1 argument.  
See 'docker pull --help'.
```

2. Do you remember what a registry is?

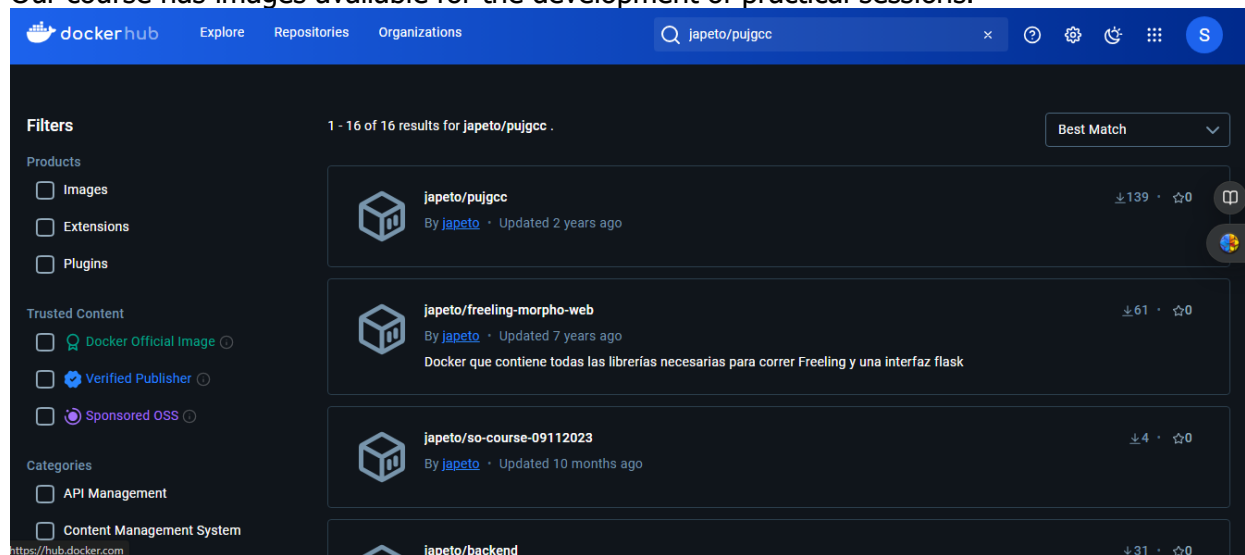
R/ En este contexto, un registro actúa como un repositorio donde se pueden subir imágenes y otros pueden descargarlas de acuerdo con la necesidad.

Download a predefined image available on the DockerHub

In a web browser, navigate to the DockerHub : <https://hub.docker.com/>

In the top search bar, type : `japeto/pujgcc` and paste a screenshot

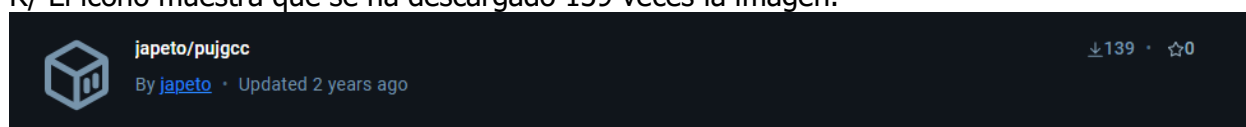
Our course has images available for the development of practical sessions.



Questions

1. How many times was the *japeto* image downloaded ?

R/ El ícono muestra que se ha descargado 139 veces la imagen.



Execute the command inside a terminal.

Paste a screenshot of result

```
docker pull japeto/pujgcc:v0.12
```

You will get an error as this image has no default tag ("latest"). So we need to specify one in the command line.

Este es el error que sale.

```
C:\Users\sebat>docker pull japeto/pujgcc:v0.12
```

What's next:

View a summary of image vulnerabilities and recommendations → [docker scout quickview japeto/pujgcc:v0.12](#)
Error response from daemon: manifest for japeto/pujgcc:v0.12 not found: manifest unknown: manifest unknown

Go to the "Tags" tab and copy the pull command of version latest

Paste a screenshot of result

```
docker pull japeto/pujgcc:v0.12
```

```
C:\Users\sebat>docker pull japeto/pujgcc:latest
```

```
latest: Pulling from japeto/pujgcc
```

```
Digest: sha256:9b3d7bbf410396d0a26e6bcffbb54e4239736d5a23ad694b03d93c26f9fc6868
```

```
Status: Image is up to date for japeto/pujgcc:latest
```

```
docker.io/japeto/pujgcc:latest
```

What's next:

View a summary of image vulnerabilities and recommendations → [docker scout quickview japeto/pujgcc:latest](#)

Question:

1. How many times do you see 'Pull complete' displayed ? Why ?

R/ Como se puede ver en la imagen no me aparece ningún "pull complete", pero me imagino que una vez por el hecho de que se completó la descarga.

Now, to be sure that the image was correctly pulled, let's see the list of all available downloaded images inside our workspace. *Paste a screenshot of result*

```
docker images
```

```
C:\Users\sebat>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
japeto/pujgcc	latest	05141310a94c	19 months ago	1.39GB

--

Question:

1. What is the size of the japeto/pujgcc image ?

R/ El tamaño de la imagen es de 1.39 GB

Perform a task using a pulled image

Among the Docker commands, we will now use the 'run' command.

Question:

1. What are the options and parameters of the 'run' command?

R/ El comando tiene al menos 1 parametros, pueden ser varios.

```
C:\Users\sebat>docker run
"docker run" requires at least 1 argument.
See 'docker run --help'.
```

Para cada opción de comando hay múltiples usos:

- Comando -a Ej: attach list
- Comando -c Ej: cpus decimal
- Comando -d Ej: detach
- Comando -e Ej: expose list
- Comando -h Ej: hostname string
- Comando -I Ej: interactive
- Comando -l Ej: label list
- Comando -m Ej: memory bytes
- Comando -p Ej: Publish list
- Comando -q Ej: quiet
- Comando -t Ej: tty
- Comando -u Ej: user string
- Comando -v Ej: volume list
- Comando -w Ej: workdir string

```
docker run --help
```

As displayed in the terminal, the description of the command is 'Run a command in a new container'.

```
C:\Users\sebat>docker run --help

Usage:  docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Create and run a new container from an image

Aliases:
    docker container run, docker run

Options:
    --add-host list                Add a custom host-to-IP mapping
                                   (host:ip)
    --annotation map              Add an annotation to the
                                   container (passed through to the
                                   OCI runtime) (default map[])
    -a, --attach list            Attach to STDIN, STDOUT or STDERR
    --blkio-weight uint16        Block IO (relative weight),
                                   between 10 and 1000, or 0 to
                                   disable (default 0)
    --blkio-weight-device list    Block IO weight (relative device
                                   weight) (default [])
    --cap-add list               Add Linux capabilities
    --cap-drop list             Drop Linux capabilities
    --cgroup-parent string       Optional parent cgroup for the
                                   container
    --cgroupns string            Cgroup namespace to use
                                   (host|private)
    --host string                Run the container in
```

Question :

1. What is the difference between an image and a container ?

R/ Una imagen es un archivo inmutable que contiene todo el software necesario para ejecutar una aplicación. Se conoce como plantillas a partir de las cuales se crean contenedores. Por otro lado, un contenedor es una instancia en ejecución de una imagen, la cual puede cambiar su estado durante la ejecución.

Now, to run the application, execute the following command:

Paste a screenshot of result

```
docker run japeto/pujgcc bash --help
```

```
C:\Users\sebat>docker run japeto/pujgcc bash --help
GNU bash, version 4.3.30(1)-release-(x86_64-pc-linux-gnu)
Usage: bash [GNU long option] [option] ...
        bash [GNU long option] [option] script-file ...
GNU long options:
  --debug
  --debugger
  --dump-po-strings
  --dump-strings
  --help
  --init-file
  --login
  --noediting
  --noprofile
  --norc
  --posix
  --rcfile
  --restricted
  --verbose
  --version
Shell options:
  -ilrsD or -c command or -O shopt_option      (invocation only)
  -abefhkmnptuvxBCHP or -o option
Type 'bash -c "help set"' for more information about shell options.
Type 'bash -c help' for more information about shell builtin commands.
Use the 'bashbug' command to report bugs.
```

Congratulations!

You just successfully downloaded and used your first Docker image!

Running *PUJGCC* without parameters was interesting as a demonstration of Docker's features. But if we want to really run *PUJGCC*, we also need to provide parameters and, most importantly, input files.

Find the paths to bind

To bind our current folder to the `/data/` folder located inside a container, we first need the absolute path of the current folder, obtained through the unix `pwd` command.

Paste a screenshot of result

```
pwd
```

```
Path
----
C:\Users\sebat\Desktop\SO_Trabajos\Actividad1
```

This path will be used in further commands through `${PWD}`.

Instead of running `ls` command to `/home/` files, we will now just list the content of the `/data/` folder inside the container but bind with the host.

Paste a screenshot of result

```
docker run japeto/pujgcc:latest ls /data
```

If nothing appears, it is normal: the folder is empty and only serves as a “*branching point*”.

```
C:\Users\sebat\Desktop\SO_Trabajos\Actividad1>docker run japeto/pujgcc:latest ls /data
ls: cannot access /data: No such file or directory
C:\Users\sebat\Desktop\SO_Trabajos\Actividad1>
```

We now have the paths of the two folders we want to bind together.

Bind a local folder into a container

To perform the folder mapping between the current folder and `/data` inside the image, the syntax is simple. *Paste a screenshot of result*

```
docker run -v ${PWD}:/data/ japeto/pujgcc:latest ls /data/
```

Question:

1. Is the displayed list the same as what is in your current folder?

```
PS C:\Users\sebat\Desktop\SO_Trabajos\Actividad1> docker run -v ${PWD}:/data/ japeto/pujgcc:latest ls /data/
data
```

Finally, we can run C on a C or C++ file located in the Data folder. Change the name of the file to any of the provided files.

Paste a screenshot of result

```
docker run -v ${PWD}:/data/ japeto/pujgcc:latest gcc /data/helloworld.c
```


Se hizo con uno de los ejercicios llamado upperCase.c

```
PS C:\Users\sebat\Desktop\S0_Trabajos\Lab-1-S0\Ejercicios> docker run -v ${PWD}:/data/ japeto/pujgcc:latest gcc /data/up
perCase.c
/data/upperCase.c: In function 'encontrarMayus':
/data/upperCase.c:14:5: error: 'for' loop initial declarations are only allowed in C99 or C11 mode
    for (int i = 0; i < MAX; i++) {
    ^
/data/upperCase.c:14:5: note: use option -std=c99, -std=gnu99, -std=c11 or -std=gnu11 to compile your code
/data/upperCase.c: In function 'main':
/data/upperCase.c:35:9: error: 'for' loop initial declarations are only allowed in C99 or C11 mode
    for (int i = 0; i < cad.cont; i++) {
    ^
PS C:\Users\sebat\Desktop\S0_Trabajos\Lab-1-S0\Ejercicios>
```

Restart and detach a container

Learn how to re-use a container where you installed something

Use the start command to restart the container created in the last exercise

Paste a screenshot of result

```
docker start -ti mycontainer /bin/bash
```

En este punto se presentó un inconveniente, ya que a la hora de ejecutar el comando aparecía que la opción “-ti” no hace parte como argumento de “start”, evidenciado en la siguiente captura.

```
PS C:\Users\sebat\Desktop\S0_Trabajos\Lab-1-S0\Ejercicios> docker start -ti mycontainer /bin/bash
unknown shorthand flag: 't' in -ti
See 'docker start --help'.
```

Este suceso se puede verificar cuando se ejecuta el comando de “Docker start --help”, el cual muestra los argumentos disponibles para usarse con la instrucción “start”, donde al fin y al cabo “-ti” no aparece

```
PS C:\Users\sebat\Desktop\S0_Trabajos\Lab-1-S0\Ejercicios> docker start --help

Usage:  docker start [OPTIONS] CONTAINER [CONTAINER...]

Start one or more stopped containers

Aliases:
  docker container start, docker start

Options:
  -a, --attach                Attach STDOUT/STDERR and forward signals
  --detach-keys string        Override the key sequence for detaching a
                              container
  -i, --interactive           Attach container's STDIN
```

Una posible solución es usar “docker run --name actividad1 -d japeto/pujgcc:latest” para inicializar el contenedor llamado “actividad1”, después para activarlo se introduce “docker start actividad1” y finalmente se escribe la siguiente instrucción que aparece en la guía, siendo “docker exec -ti actividad1 /bin/bash”. Se evidencia en la siguiente captura.

```

[ ] [ ] actividad1 8001326c53f5 japeto/pujgcc:latest Running 0% 5 minutes ago [ ] [ ] [ ]
```

```
PS C:\Users\sebat\Desktop\S0_Trabajos\Lab-1-S0\Ejercicios> docker start actividad1
actividad1
PS C:\Users\sebat\Desktop\S0_Trabajos\Lab-1-S0\Ejercicios> docker exec -ti actividad1 /bin/bash
root@8001326c53f5:/#
```

Go back to the container using the exec command instead of the run command.
Paste a screenshot of result

```
docker exec -ti mycontainer /bin/bash
```

De acuerdo con la situación comentada anteriormente, se propuso una solución la cual se presenta a continuación.

```
PS C:\Users\sebat\Desktop\S0_Trabajos\Lab-1-S0\Ejercicios> docker start actividad1
actividad1
PS C:\Users\sebat\Desktop\S0_Trabajos\Lab-1-S0\Ejercicios> docker exec -it actividad1 /bin/bash
root@8001326c53f5:/# |
```

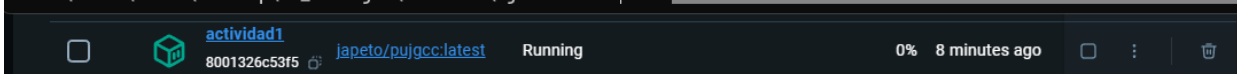
Question :

1. What happens now when you exit the container? Is it stopped?

R/ Con la instrucción "exit" uno se sale del contenedor, sin embargo, se nota que el contenedor sigue en ejecución.

```
root@8001326c53f5:/# exit
exit

What's next:
Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug actividad1
Learn more at https://docs.docker.com/go/debug-cli/
PS C:\Users\sebat\Desktop\S0_Trabajos\Lab-1-S0\Ejercicios> |
```



Check with and Paste a screenshot of result

```
docker ps -l
```

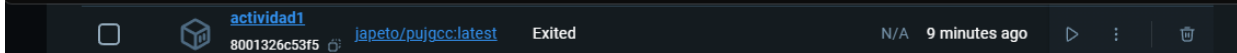
In fact, the container keeps on running. This is because re-starting a container turns it into a "detached process" running in the background. Alternatively, we could have added the -d option to the first docker run command, creating directly a detached container.

```
PS C:\Users\sebat\Desktop\S0_Trabajos\Lab-1-S0\Ejercicios> docker ps -l
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS              PORTS          NAMES
4fd3cef9c885   japeto/pujgcc:latest  "/bin/bash"            19 minutes ago  Exited (130) 18 minutes ago           naughty_poitras
PS C:\Users\sebat\Desktop\S0_Trabajos\Lab-1-S0\Ejercicios>
```

Finally, you can stop the container.

```
docker stop mycontainer
```

```
PS C:\Users\sebat\Desktop\S0_Trabajos\Lab-1-S0\Ejercicios> docker stop actividad1
actividad1
PS C:\Users\sebat\Desktop\S0_Trabajos\Lab-1-S0\Ejercicios> |
```



This is the end of the practical session. We hope you enjoyed it. Don't hesitate to ask any questions and feel free to contact us any time after the session!

List of commands

Search the available versions of an image in the Docker registry:

```
docker search
```

Pulling an image:

```
docker pull
```

Starting a container on a given image running a single command:

```
docker run -ti
```

Starting a container on a given image running a single command (detached):

```
docker run -d
```

List all containers and their status

```
docker ps -l
```

List all pulled images

```
docker images
```

Removing one local container

```
docker rm
```

Removing one local image

```
docker rmi
```

Clean all containers

```
docker rm $(docker ps -aq)
```

Clean all images (after cleaning the containers)

```
docker rmi $(docker images -aq)
```

Observations

- Deliveries must be made in teams of 4. Using a public repository on github and a pdf report
- If you do not understand the instructions for any of the activities, do not hesitate to write to jeffersonamado.pena@javerianacali.edu.co.