The _____ is responsible for creating new processes in an operating system.
Answer: operating system

The process of starting a new program or task in an OS is called _____.
Answer: process creation

When you double-click an application icon, the OS creates a _____ to run it.
Answer: new process

In Unix-based systems, the function _____ is used to create a child process.
Answer: fork()

To replace the current process image with a new program, the function _____ is used.
Answer: exec()

A process that completes its execution normally is said to _____.
Answer: exit

The OS must provide a method to _____ a process if it becomes unresponsive.
Answer: destroy

In Linux, the command used to kill a process forcefully is _____.
Answer: kill -9 <PID>

The _____ command allows a parent process to wait for its child to complete.
Answer: wait()

The purpose of a waiting interface is to allow for process _____.
Answer: synchronization

Suspending a process means to stop it from _____ temporarily.
Answer: running

To resume a suspended process, the command _____ is used in Linux.
Answer: kill -CONT <PID>

To pause a process in Linux, the command _____ can be used.
Answer: kill -STOP <PID>

Operating systems also allow changing the _____ of a process to control CPU time.
Answer: priority

The command used to set a process's priority in Linux is _____.
Answer: nice

The command used to change an already running process’s priority is _____.
Answer: renice

The interface that allows users to check how long a process has run is called _____.
Answer: status interface

The command _____ shows currently running processes and their info in Linux.
Answer: ps

The real-time process monitor in Linux that shows CPU/memory usage is _____.
Answer: top

The GUI tool in Windows that shows running processes is _____.
Answer: Task Manager

Each process has a unique identifier called _____.
Answer: Process ID (PID)

The parent of a process is identified using _____.
Answer: Parent PID (PPID)

The operating system maintains information about each process using _____.
Answer: Process Control Block (PCB)

A process that has finished execution but still has an entry in the process table is called a _____ process.
Answer: zombie

The system resource where a process is loaded and executed is _____.
Answer: memory

Process-related APIs are part of the _____ interface of the operating system.
Answer: user

Creating a new process involves loading the program and initializing _____ structures.
Answer: control

_____ allows users to view detailed statistics of running processes in GUI.
Answer: Process Explorer

A process that is temporarily stopped but still in memory is in the _____ state.
Answer: suspended

In multitasking systems, the OS uses process APIs to ensure proper _____ and control.
Answer: scheduling

The operating system must load the program's _____ and static data into memory before execution.
Answer: code

Programs initially reside on the _____ in some kind of executable format.
Answer: disk

The address space of a process includes code, static data, the heap, and the _____.
Answer: stack

Early operating systems load the entire program eagerly, while modern systems load pieces of it _____.
Answer: lazily

Lazy loading of program pieces is achieved using mechanisms like _____ and swapping.
Answer: paging

To run a program, the OS reads program bytes from _____ and places them into memory.
Answer: disk

Before running a process, the OS must allocate memory for the program’s run-time _____.
Answer: stack

The stack is used to store local variables, function parameters, and _____ addresses.
Answer: return

The OS initializes the stack with arguments like _____ and the argv array.
Answer: argc

The OS may also allocate memory for the program’s _____ to support dynamic memory allocation.
Answer: heap

In C programs, the function _____ is used to allocate memory dynamically.
Answer: malloc()

Dynamically allocated memory is released using the _____ function.
Answer: free()

The heap is initially small and can grow as the program runs and requests more _____.
Answer: memory

Data structures such as linked lists and trees are usually stored in the program's _____.
Answer: heap

The OS also sets up input/output by creating default open _____.

Answer: file descriptors

In UNIX systems, each process starts with three default file descriptors: standard input, output, and _____.
Answer: error

File descriptors allow the process to read input from the _____ and print output to the screen.
Answer: terminal

The last task of the OS in process creation is to transfer control to the process's entry point, namely the _____ function.
Answer: main()

To start execution, the OS jumps to the main() routine using a specialized _____.
Answer: mechanism

Once the OS jumps to main(), it transfers control of the _____ to the newly created process.
Answer: CPU

A program becomes a process when it is loaded from _____ into memory.
Answer: disk

The _____ stores the compiled code and static data of a program.
Answer: executable file

The OS loads code and static data into the process's _____ space.
Answer: address

_____ loading loads all parts of a program before execution.
Answer: Eager

Modern OSes prefer _____ loading to improve performance.
Answer: lazy

Lazy loading relies on memory techniques like _____ and swapping.
Answer: paging

The run-time _____ is allocated for local variables and function calls.
Answer: stack

The stack is initialized with parameters to the _____ function.
Answer: main()

The first parameter passed to main is an integer called _____.
Answer: argc

The second parameter passed to main is an array of strings called _____.

Answer: argv

The _____ is allocated for dynamically created data structures.
Answer: heap

The malloc() function in C is used to allocate memory on the _____.
Answer: heap

A _____ is a table-like structure used by the OS to track each process.
Answer: Process Control Block (PCB)

A process ID, or _____, uniquely identifies each running process.
Answer: PID

The file descriptors 0, 1, and 2 represent standard input, output, and _____.
Answer: error

The OS sets the _____ to the address of main() to start execution.
Answer: program counter

After loading and setup, the OS hands over control of the _____ to the new process.
Answer: CPU

During execution, the process may request more heap memory using the _____ system call indirectly through malloc().
Answer: brk or sbrk

Data structures like linked lists and trees are stored on the _____ segment.
Answer: heap

Arguments like argc and argv are stored in the process's _____ segment.
Answer: stack

The code section is read-only to prevent accidental modification of _____.
Answer: instructions

The OS uses the scheduler to place the process in the _____ queue.
Answer: ready

Once the process is chosen for execution, its state becomes _____.
Answer: running

The file descriptors allow interaction with I/O devices through a unified _____ interface.
Answer: file

In most systems, the memory layout includes code, static data, heap, and _____ (top-down).
Answer: stack

A. Basic Process States

A process that is currently using the CPU is said to be in the _____ state.
Answer: Running

A process that is waiting to use the CPU is in the _____ state.
Answer: Ready

A process that is waiting for an event or I/O to complete is in the _____ state.
Answer: Blocked

The three main states of a process are: Running, Ready, and _____.
Answer: Blocked

A process transitions to blocked state when it requests an operation like _____.
Answer: I/O

Once I/O is completed, a blocked process becomes _____ again.
Answer: Ready

A running process that is preempted goes to the _____ state.
Answer: Ready

A process goes from ready to running when it is _____ by the OS.
Answer: Scheduled

When a process performs I/O, it cannot continue execution and thus becomes _____.
Answer: Blocked

If a process finishes execution, it enters the _____ state.
Answer: Terminated

◇ B. State Transitions and Scheduling

The component responsible for deciding which process to run next is the _____.
Answer: Scheduler

A process goes from running to ready if its time slice is _____.
Answer: Over

From the ready state, a process can be moved to running when CPU becomes _____.
Answer: Available

A blocked process cannot be scheduled to run until the _____ it is waiting for happens.
Answer: Event

Moving a process from running to ready is known as _____.
Answer: Descheduling

Moving a process from ready to running is called _____.
Answer: Scheduling

A blocked process might be waiting for a _____ read or network packet.
Answer: Disk

A ready process has everything it needs to run except the _____.
Answer: CPU

The OS controls all process state transitions using its internal _____.
Answer: Scheduler

A blocked process uses zero CPU, allowing other processes to _____.
Answer: Run

◇ C. Multi-process Scenarios
When Process 0 becomes blocked, the OS can switch to _____.
Answer: Process 1

Running two processes with no I/O results in regular _____ switching.
Answer: CPU

When one process blocks, the OS gives CPU time to the next process in the _____ queue.
Answer: Ready

When an I/O completes, the previously blocked process moves to the _____ queue.
Answer: Ready

The goal of running another process while one is blocked is to improve CPU _____.
Answer: Utilization

Process 0 waits during I/O, so the CPU is given to _____.
Answer: Process 1

In the second example, Process 0 performs I/O and becomes _____.
Answer: Blocked

When Process 1 finishes, and Process 0 is ready, it becomes _____.
Answer: Running

The OS may not immediately switch to a process that just became ready after I/O.
Answer: Ready

Whether to switch or not is a decision taken by the _____.

Answer: Scheduler

◇ D. Conceptual Questions

A process waiting for I/O is in a _____ state.
Answer: Blocked

A process that is not waiting for I/O and not using CPU is in the _____ state.
Answer: Ready

Only one process can be in the running state on a single-core CPU at a time.
Answer: Running

A blocked process must wait for an external event such as _____ to complete.
Answer: Input/Output

Running → Blocked transition occurs due to a _____ operation.
Answer: Blocking

The OS uses state diagrams to track and manage process transitions.
Answer: State

A trace diagram helps visualize how different processes change state over time.
Answer: State

In multi-process systems, blocking one process allows the CPU to run another, avoiding _____.
Answer: Idle time

A process that is ready but not yet running must wait in a queue called the _____ queue.
Answer: Ready

A process that requests I/O is not CPU-bound but is _____-bound.
Answer: I/O

When a running process finishes, it enters the exit or _____ state.
Answer: Terminated

A running process can be interrupted and sent back to the _____ state.
Answer: Ready

The OS keeps track of each process’s current state using the PCB, which stands for _____.
Answer: Process Control Block

The purpose of the scheduler is to manage CPU _____ among all ready processes.
Answer: Time

A process in the blocked state cannot run, even if the CPU is _____.
Answer: Free

Basic Concepts
The OS uses various _____ structures to keep track of running processes.
Answer: data

The OS maintains a _____ list to manage all running and waiting processes.
Answer: process

The data structure that stores information about a process is called the _____.
Answer: Process Control Block (PCB)

In the xv6 kernel, process information is stored in a _____ proc.
Answer: struct

The process list is sometimes called the _____ list.
Answer: task

A process that has finished execution but hasn’t been cleaned is in the _____ state.
Answer: zombie

The PCB contains info like PID, process state, memory usage, and _____ descriptors.
Answer: file

A process’s current working directory is stored as an _____ pointer.
Answer: inode

In xv6, the enum used to define process states is called _____.
Answer: proc_state

The OS tracks each process’s registers in a structure called _____.
Answer: context

B. Registers and Context Switching
When a process is stopped, its registers are saved for later _____.
Answer: restoration

To resume a process, the OS places the values back into the actual _____ registers.
Answer: CPU

The process of stopping one process and starting another is called a _____ switch.
Answer: context

In xv6, registers like eip, esp, ebx, and ecx are saved in struct _____.
Answer: context

The eip register stores the instruction pointer, also known as the _____ counter.
Answer: program

The esp register is used to manage the process's _____ pointer.
Answer: stack

C. Process States in xv6
The process state UNUSED means the slot in the process table is _____.
Answer: empty

The state EMBRYO refers to a process that is being _____.
Answer: created

A SLEEPING process is blocked and waiting for an _____.
Answer: event

A RUNNABLE process is ready to run but not yet using the _____.
Answer: CPU

A process in the RUNNING state is currently being _____.
Answer: executed

When a process finishes but has not been cleaned up, it becomes a _____.
Answer: zombie

In UNIX systems, the return value 0 usually indicates a program ran _____.
Answer: successfully

A process that returns non-zero typically signals an _____.
Answer: error

After termination, a parent process calls _____ to clean up the child.
Answer: wait()

The wait() system call allows a parent to read the child's _____ code.
Answer: return

D. Memory and File Management
The pointer *mem refers to the start of process memory.
Answer: memory

The value sz represents the _____ of process memory.
Answer: size

*kstack points to the bottom of the kernel _____ for the process.
Answer: stack

Each process can open files which are tracked in an array called _____.
Answer: ofile

E. Advanced Understanding
The OS must wake up the correct process when its I/O event completes.
Answer: blocked

The field *chan is used to indicate what event a process is _____ on.
Answer: sleeping

If the field killed is non-zero, the process has been _____.
Answer: killed

The OS uses a trap frame to manage the current interrupt state of a process.
Answer: interrupt

Trapframes help handle interrupts and system calls during process execution.
Answer: exceptions

The OS cannot clean up a zombie process until the parent acknowledges it.
Answer: parent

If a parent never waits on a zombie process, it becomes an _____ process.
Answer: orphan

In UNIX-based systems, the init process adopts orphan processes to clean them.
Answer: init

OS scheduling and resource handling depend heavily on accurate process _____.
Answer: tracking

The trapframe and context are both essential for enabling safe process switching.
Answer: context

xv6 is a teaching OS, but its data structures resemble those in real OSes like _____.
Answer: Linux

Complex OSes may have more process states and fields in their _____.
Answer: process descriptor (or PCB)