CPU intro - SistOp

Lautaro Bachmann

Contents

The Abstraction: The Process **3**
The Abstraction: A Process . . . 3
process. . . . 3
machine state: . . . 3
Process API . . . 3
Create: . . . 3
Destroy: . . . 3
Wait: . . . 3
Miscellaneous Control: . . . 4
Status: . . . 4
Process Creation: A Little More Detail . . . 4
load . . . 4
run-time stack . . . 4
heap. . . . 4
other initialization tasks, . . . 4
set the stage for program execution. . . . 4
Process States . . . 5
states a process can be in at a given time. . . . 5
Running: . . . 5
Ready: . . . 5
Blocked: . . . 5
ready and running states . . . 5
a process has become blocked . . . 5
Data Structures . . . 5
process list . . . 5
register context . . . 5
other states a process can be in, . . . 5
ASIDE: KEY PROCESS TERMS . . . 6
process . . . 6
process API . . . 6
process states, . . . 6
process list . . . 6

The Abstraction: The Process

The Abstraction: A Process

process.

a process is simply a running program; at any instant in time, we can summarize a process by taking an inventory of the different pieces of the system it accesses or affects during the course of its execution.

machine state:

what a program can read or update when it is running.

memory.
Instructions lie in memory; the data that the running program reads and writes sits in memory as well. Thus the memory that the process can address (called its **address space)** is part of the process.

registers;
many instructions explicitly read or update registers and thus clearly they are important to the execution of the process.

special registers
program counter (PC) tells us which instruction of the program will execute next;

stack pointer and frame pointer are used to manage the stack for function parameters, local variables, and return addresses.

persistent storage devices
Such I/O information might include a list of the files the process currently has open.

Process API

Create:

method to create new processes.

Destroy:

an interface to destroy processes forcefully.

Wait:

wait for a process to stop running;

Miscellaneous Control:

For example, some kind of method to suspend a process and then resume it

Status:

get status information such as how long it has run for, or what state it is in.

Process Creation: A Little More Detail

load

the process of loading a program and static data from disk and place them in memory somewhere modern OSes perform the process **lazily,** by loading pieces of code or data only as they are needed during program execution.

run-time stack

Some memory must be allocated for the program's run-time stack

C programs use the stack for local variables, function parameters, and return addresses; the OS allocates this memory and gives it to the process. The OS will also likely initialize the stack with arguments; specifically, it will fill in the parameters to the main() function, i.e., argc and the argv array.

heap.

The OS may also allocate some memory for the program's heap

the heap is used for explicitly requested dynamically-allocated data; programs request such space by calling malloc() and free it explicitly by calling free().

other initialization tasks,

particularly as related to input/output (I/O).

UNIX systems,
each process by default has three open file descriptors, for standard input, output, and error; these descriptors let programs easily read input from the terminal and print output to the screen.

set the stage for program execution.

one last task: to start the program running at the entry point, namely main(). the OS transfers control of the CPU to the newly-created process, and thus the program begins its execution.

Process States

states a process can be in at a given time.

Running:

a process is running on a processor. This means it is executing instructions.

Ready:

a process is ready to run but for some reason the OS has chosen not to run it at this given moment.

Blocked:

a process has performed some kind of operation that makes it not ready to run until some other event takes place.

ready and running states

Being moved from ready to running means the process has been **scheduled;** being moved from running to ready means the process has been **descheduled.**

a process has become blocked

the OS will keep it as such until some event occurs. at that point, the process moves to the ready state again

Data Structures

process list

all processes that are ready and some additional information to track which process is currently running. The OS must also track, in some way, blocked processes;

register context

When a process is stopped, its registers will be saved to this memory location;

other states a process can be in,

initial
state that the process is in when it is being created.

final
state where it has exited but has not yet been cleaned up (in UNIX-based systems, this is called the zombie state).

This final state can be useful as it allows other processes (usually the parent that created the process) to examine the return code of the process and see if the just-finished process executed successfully

ASIDE: KEY PROCESS TERMS

process

the major OS abstraction of a running program.

the process can be described by

its state: the contents of memory in its address space, the contents of CPU registers and information about I/O

process API

consists of calls programs can make related to processes. Typically, this includes creation, destruction, and other useful calls.

process states,

running, ready to run, and blocked. Different events transition a process from one of these states to the other.

process list

contains information about all processes in the system. Each entry is found in what is sometimes called a **process control block (PCB),** which is really just a structure that contains information about a specific process.