Application Program: giải quyết một vấn đề nào đó của người dùng.

System Program: vô hình với người dùng, dùng để vận hành hệ thống.

⇨ Hệ điều hành là một system pro đóng vai trò trung gian, người dùng và chương trình ko cần biết phần cứng vật lý.

Vai trò chính của hệ điều hành:

User: cung cấp máy ảo, sử dụng thành phần máy tính một cách trừu tượng

System: quản trị, quản lý tài, cấp phát tài nguyên.

Programmer: đơn giản hóa việc lập trình

System call: là một giao diện lập trình do hệ điều hành cung cấp, cung cấp hàm giúp thao tác với hệ thống bên dưới và các tài nguyên phần cứng hay phần mềm. (Không phải phần cứng). dịch vụ hệ điều hành

Phân loại hệ điều hành

Theo lô (bath): dùng thẻ đục lổ, các hoạt động I/O làm cho cpu phải chờ. Người dùng không tương tác. Không thể khắc phục lỗi

Đa chương (MultiProgramming): Giải quyết vấn đề I/O, nạp nhiều chương trình vào hệ thống cùng một lúc. CPU chỉ thực thi 1 chương trình, khi I/O thì chạy chương trình khác. Tối ưu hóa việc sử dụng cpu

Time sharing (Multi Tasking): Nạp nhiều chương trình, các chương trình luân phiên sử dụng cpu trong một q time => thuật toán điều phối => tối ưu trải nghiệm người dùng.

Parallel OS (multi processing): sử dụng nhiều cpu, lập trình song song khó, phụ thuộc vào phần cứng.

Real Time: ràng buộc về mặt thời gian theo đúng deadline

Cứng: bắt buộc Mềm: linh hoạt =>ko nt

Hệ thống phân tán: tương tự Parallel nhưng trên nhiều hệ thống thông qua internet. Share tài nguyên. (Nhiều hệ thống, có bộ nhớ riêng != nhiều cpu có bộ nhớ chung của parallel os)

Hệ thống nhúng: Tài nguyên hạn chế, cho vài thiết bị nhất định.

Chương trình: là một tập hệt các dòng lệnh (đối tượng tĩnh)

Process: là một chương trình đang được thực thi. (đối tượng động của chương trình). Khi chương trình được nạp vào bộ nhớ chính

Thread: là một phần của process giúp process thực hiện cùng lúc nhiều phần, giảm nhẹ việc quản lý process

Thành phần của hđh

Process(thread) management: quản lý các thao tác tạo lập, hủy. liên lạc IPC, điều phối CPU, đồng bộ hóa.

Memory management: quản lý bộ nhớ

File management: quản lý cấu trúc tổ chức tập tin thư mục, cấp phát sector

I/O management: Quản lý thiết bị nhập xuất

OS structure

Simple: MS-DOS, các thành phần của một hệ điều hành gôm chung một khối. Có thể thao tác trực tiếp với bất kỳ tầng nào nó muốn. => Lỗi => ảnh hưởng toàn hệ thống.

Monolithic structure: các thành phần của os nằm trong một module duy nhất (UNIX). Nhưng cóa phân user và kernel. Có các interface trung gian => xử lý nhanh, cùng nằm trên một layer 1 lỗi => cả đám lỗi

Layer: (Multics) Chia tầng các thành phần theo layer. Từ trong ra ngoài là hard -> user 0 -> 1. Mỗi tầng sẽ gọi dịch vụ của tầng bên dưới nó => dễ bảo trì, đơn giản => phải qua các tầng trung gian.

Microkernel: (Mach) Những phần không thể thiếu của OS, quan trọng nhất sẽ đặt trong kernel. Các thành phần khác kém quan trọng thì được đẩy ra ngoài như một application => nhân nhỏ dễ quản lý bảo trì, dễ dàng mở rộng, bảo vệ, an toàn => Vấn đề hiệu suất, lõi ít gọi các thành phần ở mức user phải gửi qua kerel xử lý. => làm giải hiệu quả kernel.

Module (solaris): Chia 2 tương tự micro, phần lõi chứa các thành phần chính yếu, phần ở mức người dùng được implement được link vào một cách động và core => kernel nhỏ dễ mở rộng và cài đặt, chỉ cần implement động vào core, an toàn.

Hybird (Linux,Mac,window,android)

Process: là một đối tượng động của chương trình, file dưới dạng mã nhị phân được nạp vào bộ nhớ chính và sở hữu các tài nguyên để thực thi thì nó là một tiến trình.

Process state:

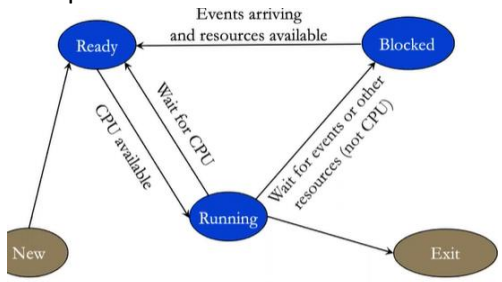
New: Hệ điều hành tự khởi động hoặc người dùng thực thi bằng hành động click. Lập trình viên khởi tạo bằng lệnh.

Ready: Sau khi new và tạo đủ tài nguyên cho tiến trình này nhưng chưa có cpu, tiến trình sẽ vào hàng đợi ready queue.

Running: sau khi được giao cpu thì chuyển trạng thái sang running. Nếu bị lấy cpu thì quay trở về hàng đợi ready queue.

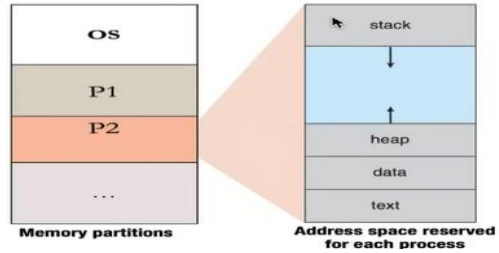
Block: Khi process chờ bất kỳ một tài nguyên nào khác ngoài cpu, hoặc sự kiện nào đó thì sẽ chuyển trạng thái sang block và vào hàng đợi waiting queue. Sau khi có được tài nguyên mong muốn thì chuyển trạng thái sang ready vào vào hàng đợi ready queue chờ giao CPU

Exit: process kết thúc



Process address space

Phân vùng bộ nhớ.



Stack: tham số của hàm, biến cục bộ trong phạm vi hàm, cấp phát và hủy cấp sẽ do hệ điều hành quản lý.

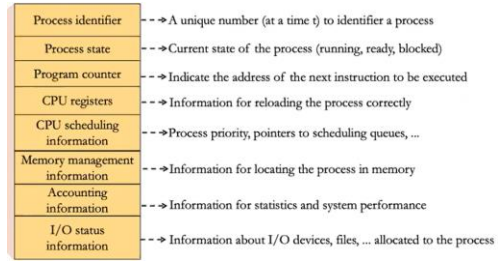
Heap: lưu các biến được cấp phát động.

Data: lưu các biến Global hay static

Text: đoạn mã nhị phân của chương trình

Process Control block:

Là cấu trúc dữ liệu đặc biệt chứa những thông tin về tiến trình (ứng với mỗi tiến trình) nằm ở phân vùng bộ nhớ của OS. OS sử dụng để quản lý hệ thống.



Context switch: chuyển đổi ngữ cản, P0 đang chạy, hđh giao cpu cho p1 bằng cách lưu trạng thái hiện thời P0 vào PCB0 (con

lệnh, thanh ghi …) sau đó load thông p1 từ PCB1 để thực thi.

3 cách tạo process: hệ thống chạy process ngầm để thực thi. Người dùng click vào chương trình. Lập trình sử dụng systemcall để tạo. tạo pid và tài nguyên, tạo PCB trong process table.

Process kết thúc: hoàn tất hoặc bị kết thúc bởi cha hoặc tiến trình hệ thống

Zombie process: một tiến trình kết thúc nhưng process cha không bắt được trạng thái kết thúc của process con => PCB chưa hủy. => dùng system call wait hoặc WaitingForSingleObject để hủy pcb

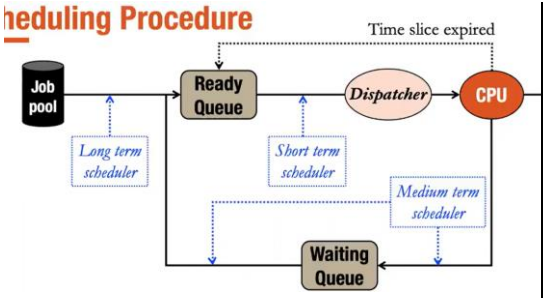
Orphan process: Cha kết thúc nhưng con chưa kết thúc => gán con cho ông nội, lên tới tiến trình root. Root sẽ quét và hủy các tiến trình mồ côi. Nếu root bị hủy thì tất cả kết thúc – cascading termination

IPC

Pipe: là cơ chế sử dụng pipe đặc biệt, chia sẽ thông tin một chiều. Writer -> Reader. 2 Loại: Ordinary pipes cha con Named pipes bất kỳ, FIFO

Share memory: Tạo một vùng share memory trong vùng nhớ riêng, process khác sẽ gắn vùng này vào không gian địa chỉ của mình. => bài toán đồng bộ.

Socket: làm một điểm cuối giao tiếp giữa các tiến trình được định danh bằng địa chỉ IP và Port number.



Scheduler:

Longterm: lựa chọn các job trong job pool và đẩy vào bộ nhớ cụ thể là ready queue

Short term: lựa chọn các process sẽ được sở hữu cpu trong ready queue.

Medium term: nhập xuất

Dispatcher: giao cpu cho process được chọn bởi short term chọn ra. (chuyển đổi ngữ cảnh)

Tiêu chí điều phối: công bằng và hiệu quả. Tối ưu hóa việc sử dụng cpu. Tối ưu hóa số lượng tiến trình hoàn tất trong một đơn vị thời gian. Giảm thời gian lưu trú, đợi và phản hồi.

Khi nào cần điều phối: process được tạo/hủy/block/bị ngắt do hệ thống phát sinh (hoàn thành hoặc hết timeslice)

FCFS – (áp dụng hệ thống theo lô bath)

-Độc quyền, đơn giản, tới trước phục vụ trước. Hiệu suất không cao, không có độ ưu tiên.

RR – (áp dụng hệ thống timesharing)

-Không độc quyền, lấy process đầu tiên để thực thi, cho mỗi tiến trình 1 đơn vị thời gian, hết q thu lại cpu và về cuối hàng đợi để chờ. => công bằng, ko có độ ưu tiên, cần xác định q phù hợp, cao thì giống FCFS, q quá ngắn thì chuyển đổi ngữ cảnh quá nhiều gây tốn chi phí

SJF

-Độc quyền, process nào ngắn có burst ngắn nhất thì thực thi, nếu có process ngắn hơn đi vào thì vẫn phải chờ cpu.

SRTN

-Không độc quyền, process nào ngắn có burst ngắn nhất thì thực thi, nếu có process ngắn hơn đi vào thì phải nhường cpu.

SJF và SRTN gây ra tình trạng chết đói.

Priority – (áp dụng timesharing)

-Sử dụng độ ưu tiên

=> Gây ra tình trạng chết đói.

=> Giải pháp: Aging để gia tăng độ ưu tiên của tiến trình theo đơn vị thời gian. Hoặc sử dụng nhiều hàng đợi với nhiều độ ưu tiên khác nhau.

Process	Group	Period	Arrival Time	CPU Burst	Priority
Fifo1	SCHED_FIFO	8	0	1	28
Fifo2	SCHED_FIFO	7	0	2	26
Fifo3	SCHED_FIFO	-	5	4	20
Fifo4	SCHED_FIFO	-	4	3	20
Rr1	SCHED_RR	-	3	3	10
Rr2	SCHED_RR	-	4	3	10
Autre	SCHED_OTHER	-	0	1	0

Queue		f1	f2	f3	f4	r1	r2	0
f1 f2	0	x						
f2	1		x					
f2	2		x					
r1	3					x		
f4 r2 r1	4				x			
f3 f4 r2 r1	5				x			
f3 f4 r2 r1	6				x			
f2	7		x					
f1 f2 r2 r1	8	x						
f2 r2 r1	9		x					
f3 r2 r1	10			x				
f3 r2 r1	11			x				
f3 r2 r1	12			x				
f3 r2 r1	13			x				
f2 r2 r1	14		x					
f2 r2 r1	15		x					
f1 r2 r1	16	x						
r2 r1	17						x	
r2 r1	18					x		
r2 r1	19						x	
r2 r1	20					x		
f2 r2	21		x					
f2 r2	22		x					
r2	23						x	
f1	24	x						
	25							x

Process	Arrival	CPU Burst	I/O		CPU Burst	I/O	
			Resource	Burst		Resource	Burst
P1	0	5	R1	2	2	R2	2
P2	2	1	R1	10	1	R1	4
P3	10	8	R2	1	-	-	-

Sơ đồ Gantt:

R2									P1→exit	----	P3→exit		
R1			P2	P2	P2	P1	P1	P2	P2→exit				
CPU	P1	P2	P1	idle	P3	P2	P3	P1	P3	P3	----		
	0	2	3	6	10	13	14	15	17	19	21	22	