

D S T Q Q S S  
D L M M J V S

## Sistemas de Computação

- Hardware CPU + memória + dispositivo E/S
- Sistemas operacionais
- Aplicações BD, automações, jogos etc
- Usuários define problemas de computação a serem resolvidos

SO

## Perspectiva da máquina virtual

↳ permite virtualmente

↳ de gerenciar recursos

↳ computador antigo rodar 3 caixas

↳ SO timeslice que permite processos que

tempos multirarefa

↳ parece que tem mais memoria do que tem

↳ Memória virtual SOBREPOSIÇÃO

↳ RAM efetiva PC não copiadade

## Perspectiva histórica

↳ Primeira geração (aprox 1945)

↳ tubo vacuo

↳ Passado

↳ Nenhum Software

↳ Operar por conexão de cabos e chaves

D S T Q O S S  
D L M M J V S

## Segunda Geração

- Transistores
- Device drivers
- 1ª linguagem de programação (Fortran)
- Monitor (leitor de cartões perfurados)
- Batch (off-line)  
não alterava a fila mesma

## Terceira geração

Multiprogramação  
pode alterar a adem da fila e  
programas

Quarta geração  
Baratear Computador  
- PC pessoal

MS Dos  
no command line

## Tempo atual Quinta geração

foco em dispositivos móveis

Computação em todos os lugares  
maior comunicação entre os dispositi-  
vos

Sistemas embarcados  
Interface Homem-Máquina revolucionária (por voz etc)

D S T Q Q S S  
D L M M J V S

## Inteligencia Artificial

Nuvem

Om PC típico

FPU MMU CPU

lo unidade de ponto flutuante

Buffer

lo dar agua para pessoas de fornecimento

Ligada

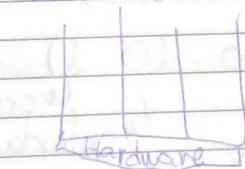
Video vamos assistindo e ele vai renderizando

9/3

Monolítico

Maquina Virtual

lo camada software com 50 diferentes



Kernel + servidores

lo Microkernel

Ex kernel + bibliotecas no funcionalidades implementadas como bibliotecas

D	S	T	Q	Q	S	S
D	L	M	M	J	V	S

Embutido na aplicação (Sistema embutido)

Estruturas

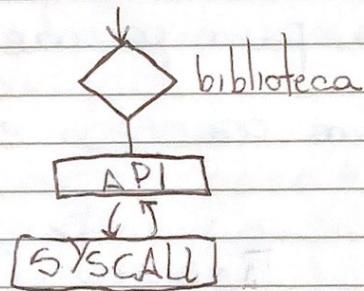
Modular

↳ replugar módulos

Orientado a Objeto

10/03/2023

APP



printf(" ");

write(fd, \*Void, Size in byte);

(↳ buffer no SO (0, 1))

(↳ saída  
↳ entrada)

#include <unistd.h>

```

int main() {
    write(1, " ", 1)
    return 0;
}
    
```

D	S	T	G	G	S	S
D	L	M	M	J	V	S

Posix

↳ interface de clocks e escalonamento

- Manipulador de sinal

↳ escuta sinal do sistema e despara a função

Alarm (s);

↳ Signal LRM

↳ propagar sinal de alarme

Signal (SIGNAL RY) <sup>sinal</sup> <sub>funcção/procedimento</sub> );

Void func (int sign) {

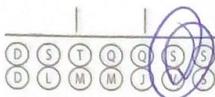
~

~

3

\* Manipulador de eventos

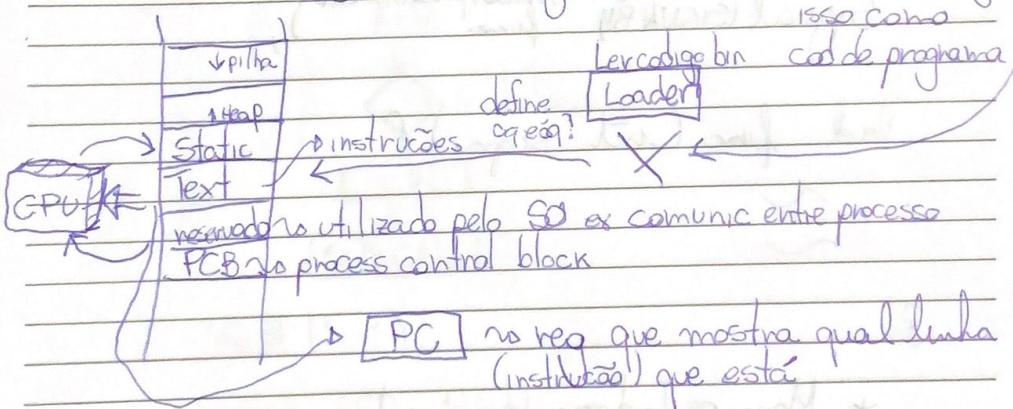
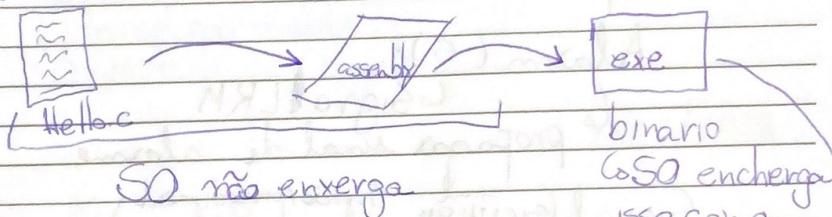
Read no scanf



## Linux: arquitetura monolítica

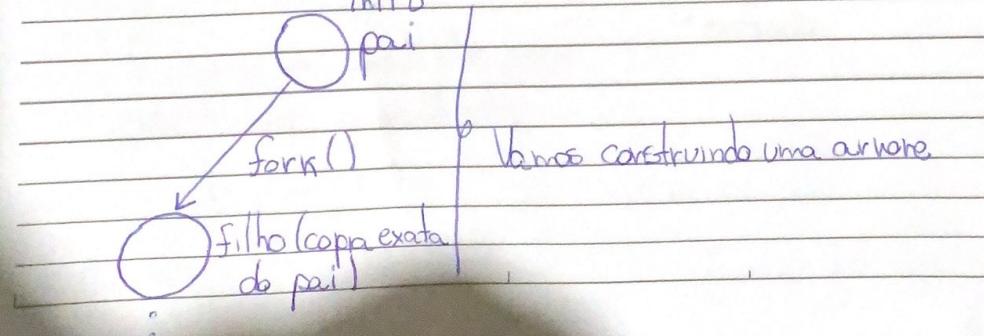
Processo: programa em execução

Código: programa quando não está em execução



Exec (serve p/ rodar o programa como quiser)  
fork() (aloca espaço na memória p/ o processo)

int d



mem

P

Prc  
que d

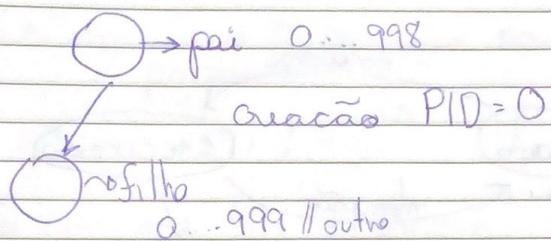
C  
seq

D	S	T	Q	Q	S	S
D	L	M	M	J	V	S

Quando um dos filhos morrem?

- tem sistema que elimina os filhos (não cria ruído mem processo zumbi)

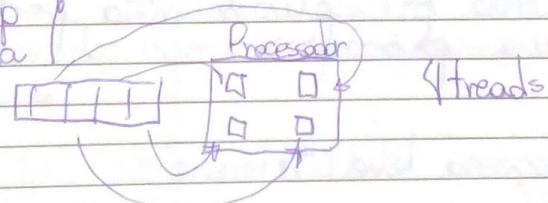
PID numero do tream



Processo não tem nada a ver com paralelismo, o que define isso são os threads

Thread

- P<sub>c</sub>
- Heap
- Pilha



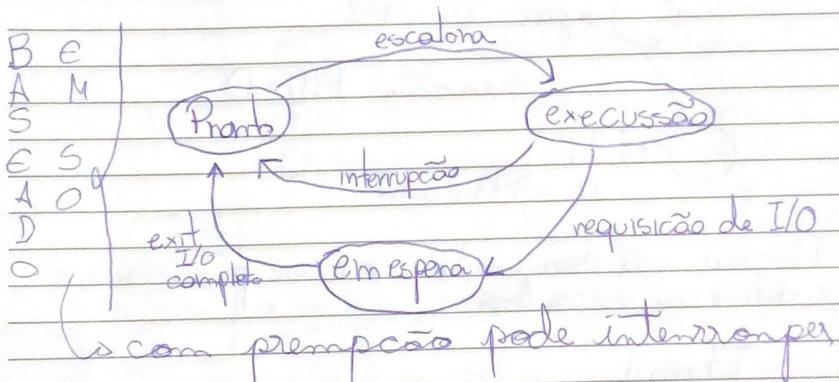
Antigamente era 1 thread rodava tudo sequencialmente.

~~WIFEXITED~~ | WIFSIGNALED  
WEXITSTATUS = WTERMSIG

D	S	T	Q	Q	S	S
D	L	M	M	J	V	S

### Estado de um processo

- Em execução
- Em espera
- Pronto



*Se não preemptivos não pode interromper um processo*

Processo tem threads

- ((o podem rodar no mesmo processo*
- ((o mesmo espaço de endereçamento*
- ((o mais rápida que processo)*

Def Processos

- Concorrente
- Independente
- Cooperantes

Mult.

CoSer

ad

p/

CoCon

Sistem

F

farr

outro praga

Execl / bin

t

D	S	T	Q	Q	S	S
D	L	M	M	J	V	S

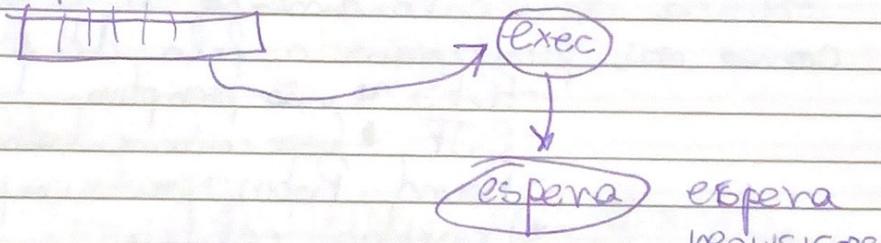
## Multi programação

↳ Sem tempo preempção interrompe e o outro tem que esperar o tempo acabar p/ começar

↳ Com menor tempo vazio com preempção  
2 processos

I/O / juntas  
CPU

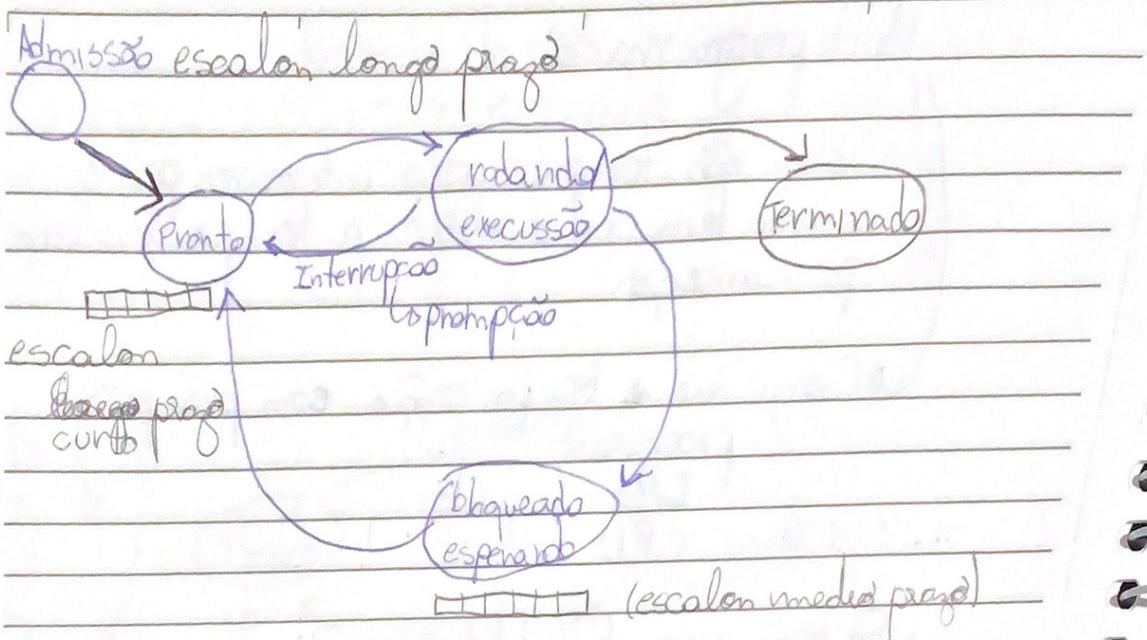
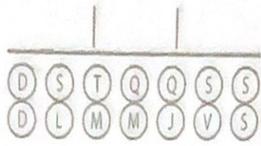
Só tem uma fila de processos  
prontos



↳ forma de informar que queremos rodar  
outro programa

Exec("bin\\ls", "-a", "-l", NULL)

↳ Caminho do binário que queremos (exe)



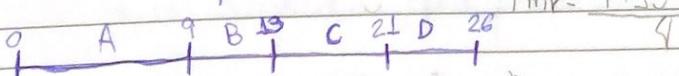
Política de escalonamento de processo  
como vai funcionar a fila de pronto

- FCFS  $\rightsquigarrow$  não preemptiva
- SJF  $\rightsquigarrow$  fazer conforme desempenho menor 1º
- Round Robin  $\rightsquigarrow$  executar um pouco de cada
- Prioridade estática.
- STF approximation

Processo	A	B	C	D	FCFS
Tempo CPU	9	13	8	5	desemp/boca
chegada	0	0	0	0	não preemptiva

Fila pronto A | B | C | D  $T_{me} = \frac{0+9+13+21}{4} = 10,75$   
 o pid menor 1º

$$T_{mr} = \frac{9+13+21+26}{4}$$



$$T_{rec} = \frac{9}{n} \quad T_{me}: \text{tempo de espera}$$

$$T_{mr} = \frac{26}{n} \quad T_{mr}: \text{tempo de resposta}$$

$$T_{exec} = \frac{26}{n} \quad T_{exec}: \text{tempo execução (variação)}$$

Tempo

$$T_{me} = \frac{10,75}{n}$$

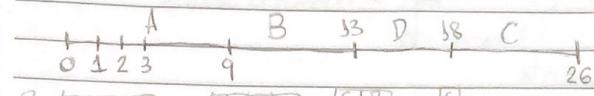
$$T_{mr} = \frac{26}{n}$$

D S T Q O S S  
D L M M J V S

	A	B	C	D
Tempo CPU	9	4	8	5
chegada	C	I	2	3

SJF  
n preempitivo

NP



Priot A B C D C

Tempo 0 [A]

Tempo 9 [C D]

Tempo 13 [B]

Tempo 18 [D]

Tempo 21 [B C]

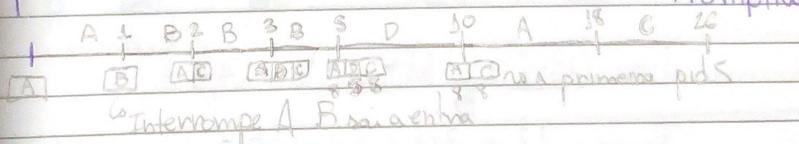
Tempo 26 [ ]

Tempo 3 [B C D]

$$Tme = 0 + (9-1) + (18-2) + (26-3) = 8,5,$$

$$Tmr = \frac{9 + 12 + 24 + 15}{4} = 15$$

P



Premptivo

\* Tempo é tempo CPU - já feito p/saber

\* A já tem 1 tempo a menos já exec but no inicio

$$Tme = (50-1) + 0 + (18-2) + (5+3) = 6,75,$$

$$Tmr = \frac{(18-0) + (5-1) + (26-2) + (50-3)}{4} = 13,25$$

D S T Q Q S S  
D L M M J V S

II previsão

$$\bar{\pi}_{i,\alpha} = \alpha \pi_i + (1-\alpha) \bar{\pi}_i \quad \text{SJF approximation}$$

iterativo preemptivo

Processo	$\bar{\pi}_i$	$t_0$	$\bar{\pi}_i t_0$	$\bar{\pi}_i t_2$	$\bar{\pi}_2$
A	1	2	2	2	1
B	1	4	2	3	3
C	1	6	3	3	2

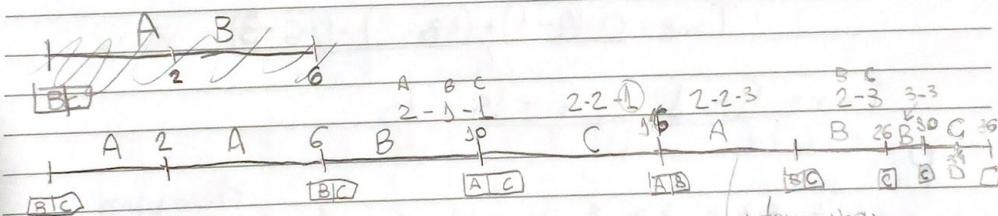
$\alpha = \frac{1}{2}$   
2 relevância dados + 1 fornecido

Prever comportamento

$$\bar{\pi}_A = \frac{1}{2} t_0 + \left(1 - \frac{1}{2}\right) \bar{\pi}_0 \approx \frac{1}{2} 2 + \left(1 - \frac{1}{2}\right) 1 = 1,5$$

$$\bar{\pi}_{B,C} = \frac{1}{2} 4 + \left(1 - \frac{1}{2}\right) 1 = 2,5 \quad \text{não considera } A$$

$$\bar{\pi}_{B,C} = \frac{1}{2} 6 + \left(1 - \frac{1}{2}\right) 1 = 3,5$$



$$Tme = (16-6) + (18) + (24) = 173 \text{ ut}$$

$$Tmr = 22 + 30 + 36 = 88 \text{ ut}$$

$$\frac{3}{36} = \text{press/ut}$$