

Processos e threads

Guilherme Henrique de Souza Nakahata

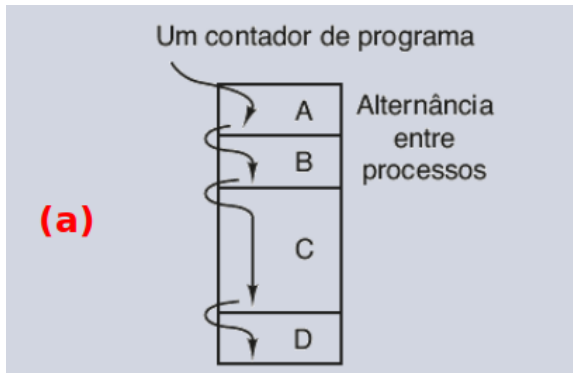
Universidade Estadual do Paraná - Unespar

31 de Março de 2023

- O conceito mais central em qualquer sistema operacional é o **processo**: uma abstração de um programa em execução.
- Um processo é apenas uma instância de um programa em execução, incluindo os valores atuais do contador do programa, registradores e variáveis.
- Processos podem ser criados e terminados dinamicamente.
- Cada processo tem seu próprio espaço de endereçamento.

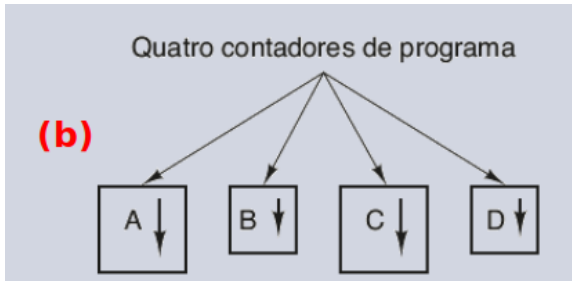
O modelo de processo

- Na figura (a) vemos um computador multiprogramando quatro programas na memória.



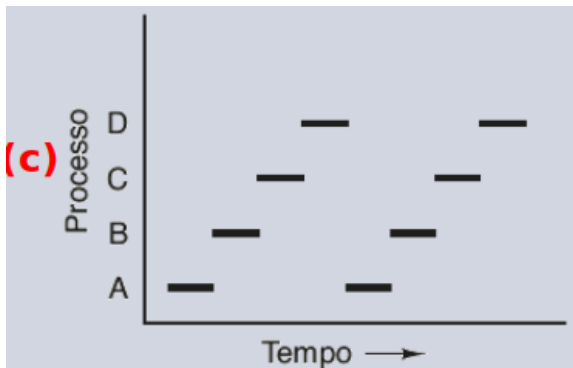
O modelo de processo

- Na figura (b) vemos quatro processos, cada um com seu próprio fluxo de controle e sendo executado independente dos outros



O modelo de processo

- Na figura (c) vemos que, analisados durante um intervalo longo o suficiente, todos os processos tiveram progresso, mas a qualquer dado instante apenas um está sendo de fato executado.



- Sistemas operacionais precisam de alguma maneira criar processos.
- Quatro eventos principais fazem com que os processos sejam criados:
 - ① Inicialização do sistema.
 - ② Execução de uma chamada de sistema de criação de processo por um processo em execução.
 - ③ Solicitação de um usuário para criar um novo processo.
 - ④ Início de uma tarefa em lote.

- Após um processo ter sido criado, ele começa a ser executado e realiza qualquer que seja o seu trabalho;
- Cedo ou tarde, o novo processo terminará, normalmente devido a uma das condições a seguir:
 - ① Saída normal (voluntária).
 - ② Erro fatal (involuntário).
 - ③ Saída por erro (voluntária).
 - ④ Morto por outro processo (involuntário).

Término de processos

```
0[|||||] 27.4% 4[||||] 7.6%
1[|||||] 17.5% 5[|] 1.3%
2[||||] 5.8% 6[|||||] 10.8%
3[|||||] 18.9% 7[|||||] 34.4%
Mem[|||||] 2.90G/15.5G Tasks: 135, 765 thr; 1 running
Swp[|||||] 0K/2.00G Load average: 0.70 0.77 0.75
Uptime: 00:34:29
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPUS	MEM%	TIME+	Command
1737	gulherne	20	0	1450M	109M	70284	S	12.8	0.7	1:58.01	/usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthority -nolisten tcp
1976	gulherne	20	0	5304M	255M	119M	S	11.1	1.6	1:36.98	/usr/bin/gnome-shell
7902	gulherne	20	0	878M	54380	41400	S	9.9	0.3	0:00.72	/usr/libexec/gnome-terminal-server
3433	gulherne	20	0	12.7G	458M	190M	S	7.6	2.9	6:45.10	/snap/firefox/2487/usr/lib/firefox/firefox
1813	gulherne	20	0	1450M	109M	70284	S	3.5	0.7	0:08.88	/usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthority -nolisten tcp
7928	gulherne	20	0	20820	5952	3664	R	3.5	0.0	0:00.32	htop
4254	gulherne	20	0	2704M	270M	99M	S	2.3	1.7	0:28.60	/snap/firefox/2487/usr/lib/firefox/firefox -contentproc -childID 3 -isForBrowser -pre
772	systemd-o	20	0	14020	6160	5368	S	0.6	0.0	0:03.88	/lib/systemd/systemd-oomd
821	avahi	20	0	1612	4996	3692	S	0.6	0.0	0:31.98	avahi-daemon: running [X510UAR.local]
1330	root	20	0	1352M	12000	10496	S	0.6	0.1	0:03.73	/opt/teamviewer/tv_bin/teamviewer -d
2334	gulherne	20	0	5127M	316M	89704	S	0.6	2.0	0:24.82	/home/gulherne/.dropbox-dist/dropbox.lnx.x86_64-170.4.5895/dropbox
3608	gulherne	20	0	12.7G	458M	190M	S	0.6	2.9	0:00.32	/snap/firefox/2487/usr/lib/firefox/firefox
3662	gulherne	20	0	12.7G	458M	190M	S	0.6	2.9	0:10.40	/snap/firefox/2487/usr/lib/firefox/firefox
3927	gulherne	20	0	2688M	188M	76824	S	0.6	1.2	0:39.80	/snap/firefox/2487/usr/lib/firefox/firefox -contentproc -childID 2 -isForBrowser -pre
4263	gulherne	20	0	2704M	270M	99M	S	0.6	1.7	0:01.01	/snap/firefox/2487/usr/lib/firefox/firefox -contentproc -childID 3 -isForBrowser -pre
4284	gulherne	20	0	2704M	270M	99M	S	0.6	1.7	0:00.55	/snap/firefox/2487/usr/lib/firefox/firefox -contentproc -childID 3 -isForBrowser -pre
1	root	20	0	162M	12004	8224	S	0.0	0.1	0:02.29	/sbin/init splash
308	root	19	1	100M	58984	57244	S	0.0	0.4	0:00.92	/lib/systemd/systemd-journald
361	root	20	0	27488	7736	4588	S	0.0	0.0	0:00.76	/lib/systemd/systemd-udev
773	systemd-r	20	0	25524	13892	9552	S	0.0	0.1	0:01.41	/lib/systemd/systemd-resolved
774	systemd-t	20	0	89380	6652	5796	S	0.0	0.0	0:00.15	/lib/systemd/systemd-timesyncd
789	systemd-t	20	0	89380	6652	5796	S	0.0	0.0	0:00.00	/lib/systemd/systemd-timesyncd
817	root	20	0	243M	7968	6820	S	0.0	0.0	0:00.20	/usr/libexec/accounts-daemon
818	root	20	0	2812	1132	1048	S	0.0	0.0	0:00.14	/usr/sbin/acpid
822	root	20	0	10624	5088	4696	S	0.0	0.0	0:00.12	/usr/lib/bluetooth/bluetoothd
823	root	20	0	18148	3824	2768	S	0.0	0.0	0:00.00	/usr/sbin/cron -f -P
824	messagebu	20	0	11292	7084	4132	S	0.0	0.0	0:03.94	@dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --

F1 Help F2 Setup F3 Search F4 Alter F5 Free F6 Sort By F7 Vibe F8 Nice F9 Kill F10 Quit

Término de processos

```

0[|||||] 10.2% 4[|] 5.7%
1[|||||] 10.5% 5[|] 3.4%
2[|||||] 7.0% 6[|] 3.5%
3[|||||] 6.0% 7[|] 1.9%
Mem[|||||] 2.92G/15.5G Tasks: 136, 770 thr; 1 running
Swp[|||||] 0K/2.00G Load average: 0.54 0.74 0.73
Uptime: 00:34:45

Send signals: PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
0 Cancel 821 avahi 20 0 8612 4996 3692 S 3.4 0.0 0:32.32 avahi-daemon: running [X510UAR.local]
1 SIGHUP 1737 guilherme 20 0 1454M 109M 70284 S 5.0 0.7 1:59.30 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthority
2 SIGINT 3433 guilherme 20 0 12.7G 457M 190M S 1.5 2.9 0:45.43 /snap/firefox/2487/usr/lib/firefox/firefox
3 SIGQUIT 4254 guilherme 20 0 2704M 266M 99M S 2.6 1.7 0:28.98 /snap/firefox/2487/usr/lib/firefox/firefox -contentproc -childID 3 -ls
4 SIGILL 7928 guilherme 20 0 20820 5952 3664 R 1.9 0.0 0:00.76 http
5 SIGTRAP 1976 guilherme 20 0 5325M 256M 119M S 4.1 1.6 1:39.00 /usr/bin/gnome-shell
6 SIGABRT 3596 guilherme 20 0 12.7G 457M 190M S 0.8 2.9 0:11.96 /snap/firefox/2487/usr/lib/firefox/firefox
7 SIGIOT 7962 guilherme 20 0 879M 55048 41400 S 4.1 0.3 0:01.26 /usr/libexec/gnome-terminal-server
8 SIGBUS 772 systemd-u 20 0 14828 6160 1368 S 0.4 0.0 0:03.91 /lib/systemd/systemd-oomd
9 SIGFPE 1813 guilherme 20 0 1454M 109M 70284 S 0.9 0.7 0:09.10 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthority
10 SIGKILL 2334 guilherme 20 0 5127M 316M 89784 S 0.8 2.0 0:24.89 /home/guilherme/.dropbox-dist/dropbox-lnx.x86_64-170.4.5895/dropbox
11 SIGUSR1 2361 guilherme 20 0 5127M 316M 89784 S 0.8 2.0 0:09.14 /home/guilherme/.dropbox-dist/dropbox-lnx.x86_64-170.4.5895/dropbox
12 SIGSEGV 3662 guilherme 20 0 12.7G 457M 190M S 1.1 2.9 0:10.56 /snap/firefox/2487/usr/lib/firefox/firefox
13 SIGUSR2 4263 guilherme 20 0 2704M 266M 99M S 0.4 1.7 0:01.11 /snap/firefox/2487/usr/lib/firefox/firefox -contentproc -childID 3 -ls
14 SIGPIPE 1 root 20 0 102M 12004 8224 S 0.0 0.1 0:02.30 /sbin/init splash
15 SIGALRM 308 root 19 -1 100M 59000 57360 S 0.0 0.4 0:00.92 /lib/systemd/systemd-journald
16 SIGTERM 361 root 20 0 27488 7736 4588 S 0.0 0.0 0:00.78 /lib/systemd/systemd-udev
17 SIGSTKFLT 773 systemd-r 20 0 25524 13892 9552 S 0.0 0.1 0:01.41 /lib/systemd/systemd-resolved
18 SIGCHLD 774 systemd-t 20 0 89380 6652 5796 S 0.0 0.0 0:00.15 /lib/systemd/systemd-timesyncd
19 SIGCONT 789 systemd-t 20 0 89380 6652 5796 S 0.0 0.0 0:00.00 /lib/systemd/systemd-timesyncd
20 SIGSTOP 817 root 20 0 243M 7968 6828 S 0.0 0.0 0:00.20 /usr/libexec/accounts-daemon
21 SIGTSTP 818 root 20 0 2812 1132 1048 S 0.0 0.0 0:00.14 /usr/sbin/acpid
22 SIGTTIN 822 root 20 0 10624 5088 4696 S 0.0 0.0 0:00.12 /usr/lib/bluetooth/bluetoothd
23 SIGTTOU 823 root 20 0 18148 3024 2768 S 0.0 0.0 0:00.00 /usr/sbin/cron -f -P
24 SIGURG 824 messagebu 20 0 11292 7084 4132 S 0.0 0.0 0:03.95 dbus-daemon --system --address=systemd: --nofork --nopidfile --system
25 SIGXCPU 826 root 20 0 263M 19248 15948 S 0.0 0.1 0:04.85 /usr/sbin/NetworkManager --no-daemon
26 SIGXFSZ 832 root 20 0 82768 3844 3484 S 0.0 0.0 0:00.27 /usr/sbin/irqbalance --foreground
EnterSend EscCancel

```

Término de processos

Send signal:

0 Cancel

1 SIGHUP

2 SIGINT

3 SIGQUIT

4 SIGILL

5 SIGTRAP

6 SIGABRT

6 SIGIOT

7 SIGBUS

8 SIGFPE

9 SIGKILL

10 SIGUSR1

11 SIGSEGV

12 SIGUSR2

13 SIGPIPE

14 SIGALRM

15 SIGTERM

16 SIGSTKFLT

17 SIGCHLD

18 SIGCONT

19 SIGSTOP

20 SIGTSTP

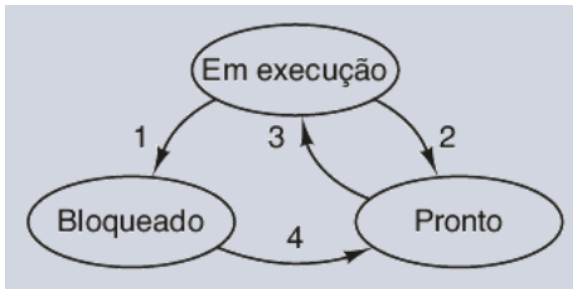
Hierarquias de processos

- Quando um processo cria outro, o processo pai e o processo filho continuam a ser associados de certas maneiras.
- O processo filho pode em si criar mais processos, formando uma hierarquia de processos.

- Embora cada processo seja uma entidade independente, com seu próprio contador de programa e estado interno, processos muitas vezes precisam interagir entre si.
- Um processo pode gerar alguma saída que outro processo usa como entrada.

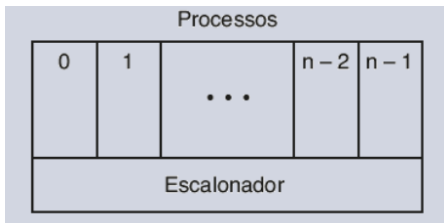
Estados de processos

- Três estados de um processo:
 - 1. Em execução.
 - 2. Pronto.
 - 3. Bloqueado.



Estados de processos

- O nível mais baixo de um sistema operacional estruturado em processos controla interrupções e escalonamento.
- Acima desse nível estão processos sequenciais.
- Todo o tratamento de interrupções e detalhes sobre o início e parada de processos estão ocultos no escalonador.

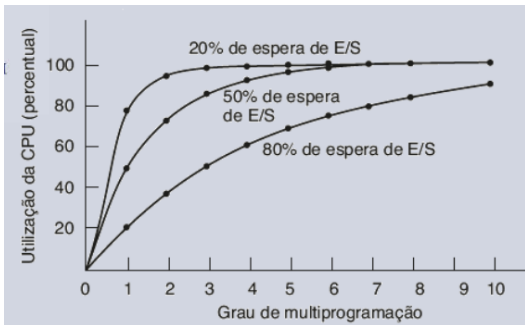


Implementação de processos

- Para implementar o modelo de processos, o sistema operacional mantém uma tabela chamada de **tabela de processos**, com uma entrada para cada um deles.
- Um processo pode ser interrompido milhares de vezes durante sua execução.
- Sempre retornando para o estado em que se encontrava antes de ser interrompido.

Modelando a multiprogramação

- Quando a multiprogramação é usada.
- A utilização da CPU pode ser aperfeiçoada.



Exemplo

- Para algumas aplicações é útil ter múltiplos threads de controle dentro de um único processo.
- Esses threads são escalonados independentemente e cada um tem sua própria pilha.
- Todas as threads em um processo compartilham de um espaço de endereçamento comum.
- Threads podem ser implementados no espaço do usuário ou no núcleo.

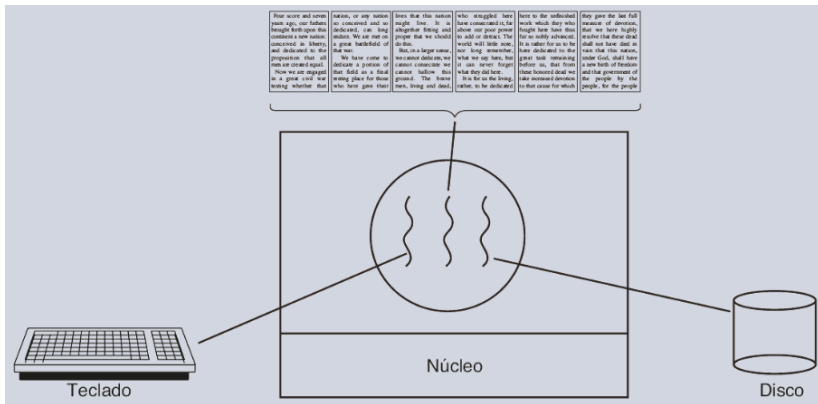
Threads

```
void IniciaThreadsGVNS(FuncaoObjetivo funcaoObjetivo, bool debug, Solucao solucaoInicial, Parametro parametro, float porcentagem, int epocas, int qtdCamadas){  
    vector<thread> threads;  
  
    // for(int i = 0; i < thread::hardware_concurrency(); i++){  
    for(int i = 0; i < 10; i++){  
        threads.push_back(thread(IniciaGVNS, funcaoObjetivo, debug, solucaoInicial, parametro, porcentagem, epocas, i, qtdCamadas));  
    }  
  
    for(int i = 0; i < threads.size(); i++){  
        threads[i].join();  
    }  
}
```

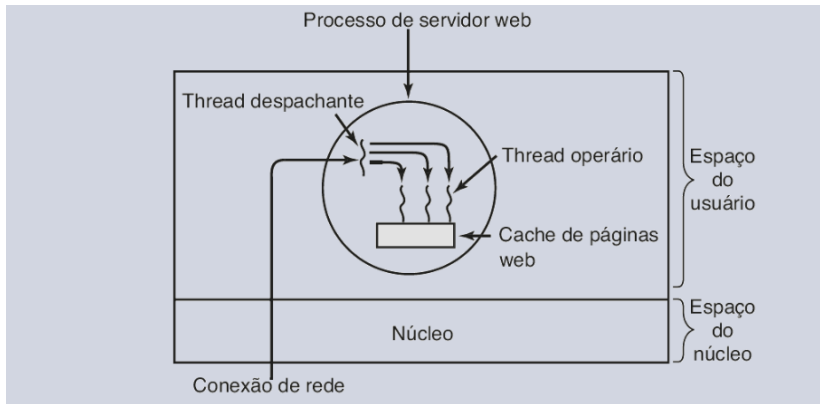
Modelos para construir um servidor

Modelo	Características
Threads	Paralelismo, chamadas de sistema bloqueantes
Processo monothread	Não paralelismo, chamadas de sistema bloqueantes
Máquina de estados finitos	Paralelismo, chamadas não bloqueantes, interrupções

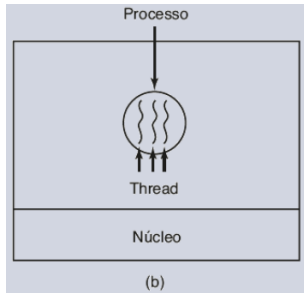
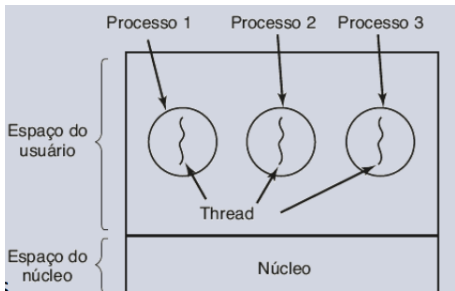
Um processador de texto com três threads



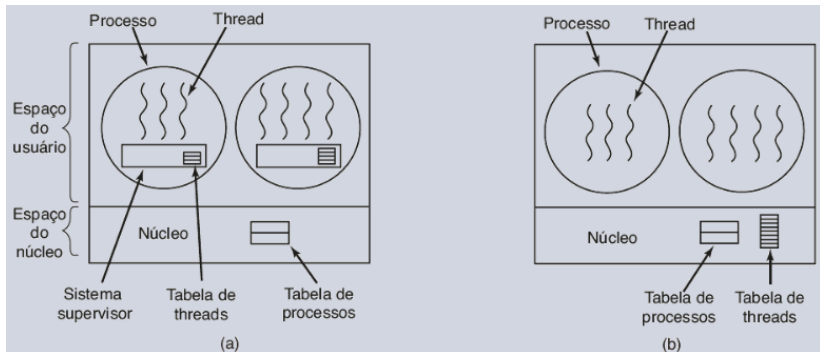
Um servidor web multithread



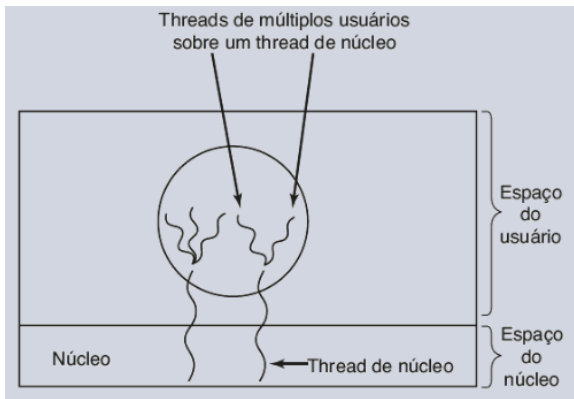
O modelo de thread clássico



Implementando threads no espaço e no núcleo

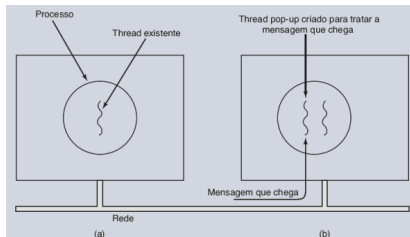


Implementações híbridas



Threads pop-up

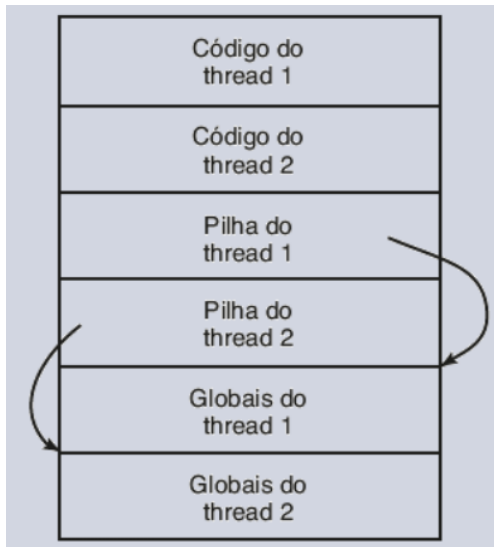
- Threads costumam ser úteis em sistemas distribuídos.
- A abordagem tradicional é ter um processo ou thread que esteja bloqueado em uma chamada de sistema recebe esperando pela mensagem que chega.
- Quando uma mensagem chega, ela é aceita, aberta, seu conteúdo examinado e processada.
- Com a chegada de uma mensagem o sistema cria uma nova thread para lidar com a mensagem. Essa thread é chamado de thread pop-up.



Convertendo código de um thread em código multithread

- Muitos programas existentes foram escritos para processos monothread. Convertê-los para multithreading é muito mais complicado do que pode parecer em um primeiro momento.
- O código de um thread em geral consiste em múltiplas rotinas, exatamente como um processo.
- Essas rotinas podem ter variáveis locais, variáveis globais e parâmetros.
- Variáveis locais e de parâmetros não causam problema algum, mas variáveis que são globais para um thread, mas não globais para o programa inteiro, são um problema.

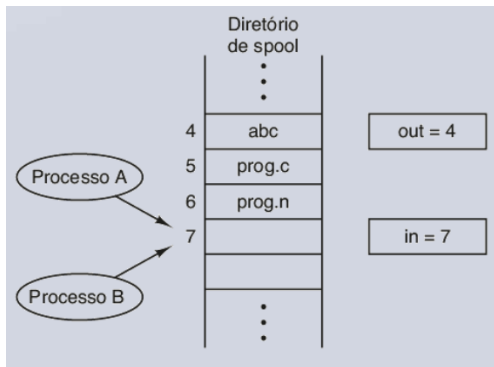
Threads podem ter variáveis globais individuais



- Processos podem comunicar-se uns com os outros usando primitivas de comunicação entre processos.
- Por exemplo: semáforos, monitores ou mensagens.
- Essas primitivas são usadas para assegurar que jamais dois processos estejam em suas regiões críticas ao mesmo tempo.

Condições de corrida

- Em alguns sistemas operacionais, processos que estão trabalhando juntos podem compartilhar de alguma memória comum que cada um pode ler e escrever.



- **Como evitar as condições de corrida?**
- A chave para evitar problemas é encontrar alguma maneira de proibir mais de um processo de ler e escrever os dados compartilhados ao mesmo tempo.
- O que precisamos é de **exclusão mútua**, isto é, alguma maneira de se certificar de que se um processo está usando um arquivo ou variável compartilhados, os outros serão impedidos de realizar a mesma coisa.

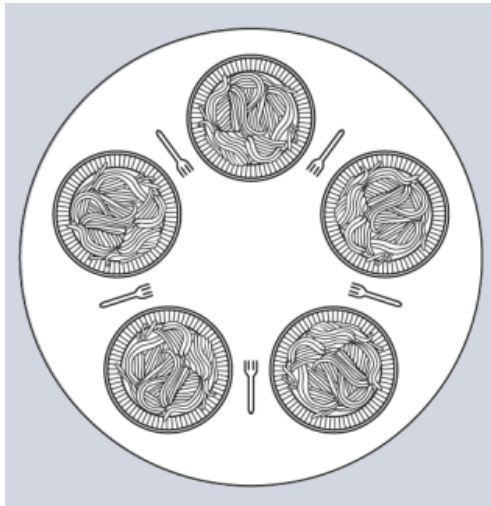


- Quando um computador é multiprogramado, ele frequentemente tem múltiplos processos ou threads competindo pela CPU ao mesmo tempo.
- Essa situação ocorre sempre que dois ou mais deles estão simultaneamente no estado pronto.
- Se apenas uma CPU está disponível, uma escolha precisa ser feita sobre qual processo será executado em seguida.
- A parte do sistema operacional que faz a escolha é chamada de escalonador, e o algoritmo que ele usa é chamado de algoritmo de escalonamento.

- **O problema do jantar dos filósofos**
- Cinco filósofos estão sentados em torno de uma mesa circular. Cada filósofo tem um prato de espaguete. O espaguete é tão escorregadio que um filósofo precisa de dois garfos para comê-lo. Entre cada par de pratos há um garfo.
- Quando um filósofo fica suficientemente faminto, ele tenta pegar seus garfos à esquerda e à direita, um de cada vez, não importa a ordem. Se for bem-sucedido em pegar dois garfos, ele come por um tempo, então larga os garfos e continua a pensar.

Problemas clássicos de IPC

- A questão fundamental é: você consegue escrever um programa para cada filósofo que faça o que deve fazer e jamais fique travado?



- **O problema dos leitores e escritores**
- Imagine, por exemplo, um sistema de reservas de uma companhia aérea, com muitos processos competindo entre si desejando ler e escrever.
- É aceitável ter múltiplos processos lendo o banco de dados ao mesmo tempo, mas se um processo está atualizando o banco de dados, nenhum outro pode ter acesso, nem mesmo os leitores.

- Processos, threads e escalonamento, não são mais os tópicos quentes de pesquisa que já foram um dia. A pesquisa seguiu para tópicos como **gerenciamento de energia, virtualização, nuvens e segurança.**

- O escalonamento de dispositivos móveis em busca da eficiência energética (YUAN e NAHRSTEDT, 2006), escalonamento com tecnologia hyperthreading (BULPIN e PRATT, 2005) e escalonamento bias-aware (KOUFATY, 2010).
- Com cada vez mais computação em smartphones com restrições de energia, alguns pesquisadores propõem migrar o processo para um servidor mais potente na nuvem, quando isso for útil (GORDON et al, 2012).
- No entanto, poucos projetistas de sistemas andam preocupados com a falta de um algoritmo de escalonamento de threads decente, então esse tipo de pesquisa parece ser mais um interesse de pesquisadores do que uma demanda de projetistas.

Obrigado! Dúvidas?

Guilherme Henrique de Souza Nakahata

guilhermenakahata@gmail.com

<https://github.com/GuilhermeNakahata/UNESPAR-2023>