

# SISTEMAS OPERACIONAIS



**Professor Fábio Angelo**  
**E-mail: [fabio.angelo@unisul.br](mailto:fabio.angelo@unisul.br)**

# CONVERSANDO SOBRE AS RESPOSTAS RECEBIDAS... (GRUPO VIEIRA)

1. Defina o conceito de processo.

R: Pode ser definido como sendo o conjunto necessário de informações para que o sistema operacional implemente a concorrência de programas.

2. Por que o conceito de processo é tão importante no projeto de sistemas multiprogramáveis?

R: Nos sistemas multiprogramáveis os processos são executados concorrentemente, compartilhando o uso do processador e memória principal.

# CONVERSANDO SOBRE AS RESPOSTAS RECEBIDAS... (GRUPO ANDRÉ)

3. É possível que um programa execute no contexto de um processo e não execute no contexto de um outro? Por quê?


R: Sim. Dependendo do contexto pode não haver disponibilidade, por exemplo, de recursos de hardware, em um contexto tenha memória ram suficiente e no outro não, como também recursos de software, onde ambos possuem espaço de disco mas um dos usuários possuem limitação de utilização. Ou seja, dependendo dos recursos e das permissões um programa execute em um contexto e em outro não

4. Quais partes compõem um processo?

R: Contexto de software, contexto de hardware e espaço de endereçamento

# CONVERSANDO SOBRE AS RESPOSTAS RECEBIDAS... (GRUPO VIEIRA)

5. O que é o contexto de hardware de um processo e como é a implementação da troca de contexto?

R: Armazena o conteúdo dos registradores gerais da UCP, além dos registradores de uso específico, como program counter (PC), stack pointer (SP) e registrador de status. A operação se resume em substituir o contexto de hardware de um processo pelo de outro, **conforme mostra a figura ao lado.** (???) 

6. Qual a função do contexto de software? Exemplifique cada grupo de informação.

R: Nele é especificado os limites e características dos recursos que podem ser alocados pelo processo. Identificação: PID; Quotas: Número máximo de arquivos abertos simultaneamente; Privilégios: Usuário "user" e usuário administrador (root).

# CONVERSANDO SOBRE AS RESPOSTAS RECEBIDAS... (GRUPO ANDRÉ)

7. O que é o espaço de endereçamento de um processo?

R: É a área de memória pertencente ao processo onde instruções e dados do programa são armazenados para execução. Cada processo possui seu próprio espaço de endereçamento, que deve ser devidamente protegido do acesso dos demais processos

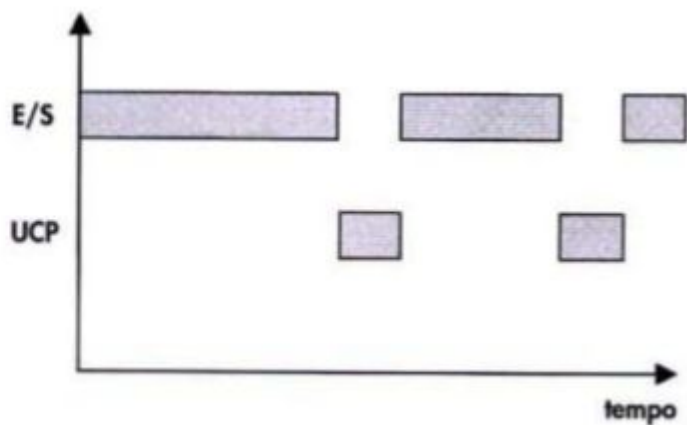
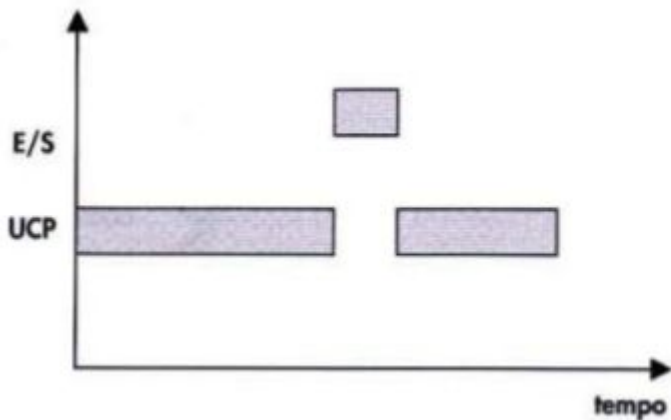
8. Como o sistema operacional implementa o conceito de processo? Qual a estrutura de dados indicada para organizar os diversos processos na memória principal?

R: Através de uma estrutura de dados chamada bloco de controle de processo (PCB).

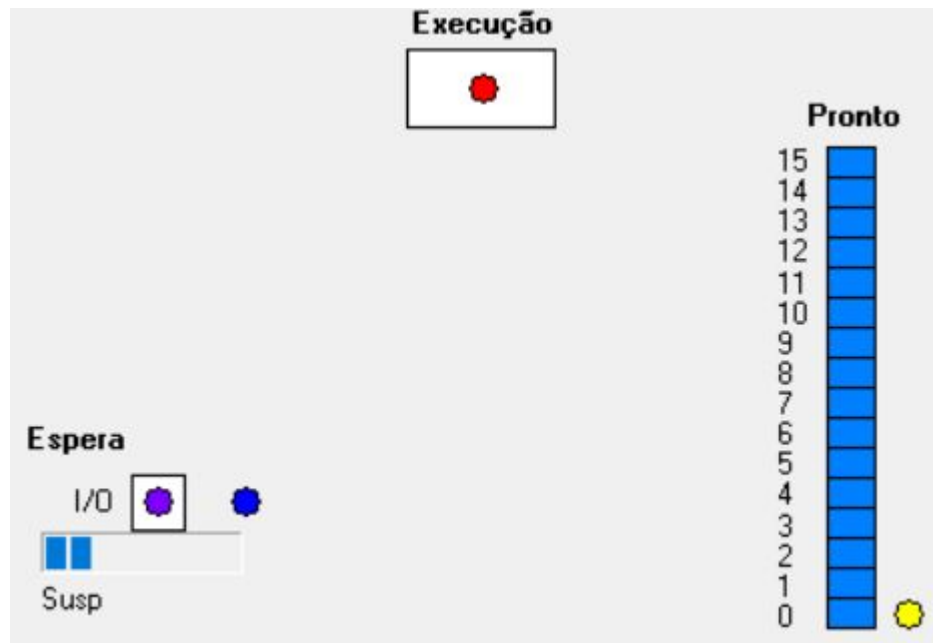
Estrutura de Ponteiros -> Estado do processo -> Nome do processo -> Prioridade do processo

# PROCESSOS I/O BOUND E CPU BOUND





- Classificação usada para terminar os processos que consomem mais recursos de CPU ou I/O (E/S);
- A identificação desse comportamento se dá pelo tempo em que processo passa no estado de “execução” ou “espera”;



# PROCESSOS I/O BOUND E CPU BOUND

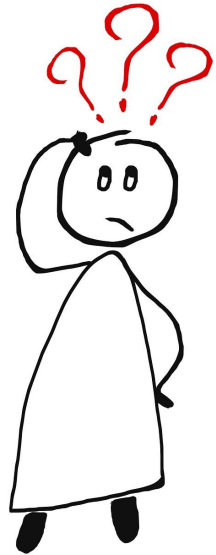


**O que está divergente  
nas imagens?**

Cor	PID	Prio	Estado	Temp UCP	Frames
	427650	0	Execução	72	5
	435066	0	I/O	8	5
	444314	0	Pronto	7	5
	504379	0	I/O	9	5

# UMA DÚVIDA...

Os programas muito interativos, que frequentemente carecem de respostas dos usuários para seguir execução, poderiam ser considerados I/O Bound?





# PROCESSOS FOREGROUND E BACKGROUND

- Um processo possui sempre pelo menos dois canais de comunicação associados a sua estrutura, sendo entrada (Input) ou saída (Output);
- Um processo foreground é aquele que permite a comunicação direta do usuário com o processo durante o seu processamento;
- Um processo background é aquele onde não existe a comunicação com o usuário durante o seu processamento.

# PROCESSOS FOREGROUND E BACKGROUND

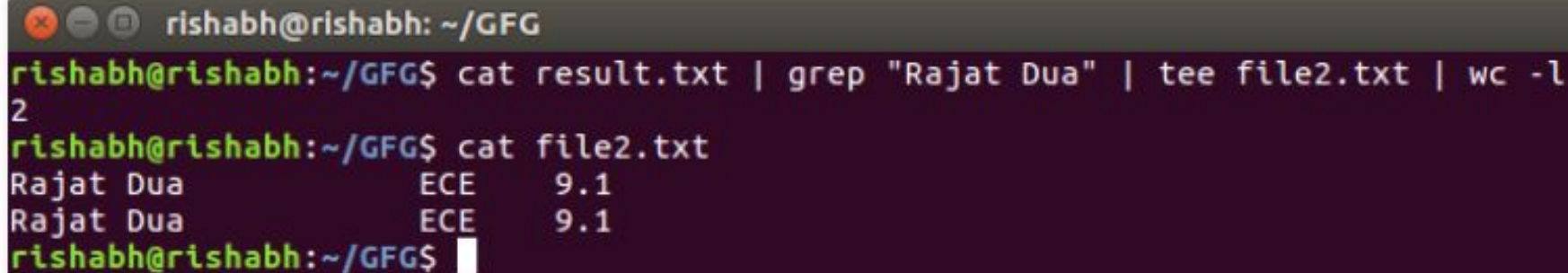


Quais exemplos de processos de cada tipo?

# PROCESSOS ENCADEADOS

- É possível associar o canal de saída de um processo ao canal de entrada de um outro processo. Neste caso dizemos que existe um pipe ligando os mesmos.

```
$ cat result.txt | grep "Rajat Dua" | tee file2.txt | wc -l
```



A terminal window with a dark background and light-colored text. The window title is 'rishabh@rishabh: ~/GFG'. The prompt is 'rishabh@rishabh:~/GFG\$'. The command entered is 'cat result.txt | grep "Rajat Dua" | tee file2.txt | wc -l'. The output is '2'. The prompt is 'rishabh@rishabh:~/GFG\$'. The command entered is 'cat file2.txt'. The output is 'Rajat Dua ECE 9.1' followed by 'Rajat Dua ECE 9.1'. The prompt is 'rishabh@rishabh:~/GFG\$' followed by a cursor.

```
rishabh@rishabh: ~/GFG
rishabh@rishabh:~/GFG$ cat result.txt | grep "Rajat Dua" | tee file2.txt | wc -l
2
rishabh@rishabh:~/GFG$ cat file2.txt
Rajat Dua      ECE      9.1
Rajat Dua      ECE      9.1
rishabh@rishabh:~/GFG$
```

# CRIAÇÃO DE PROCESSOS

- Um processo pode ser criado de diversas maneiras. Seguem as três principais formas de criação de processos:
  - Logon interativo;
  - Via linguagem de comando;
  - Via rotina do Sistema Operacional.

# CRIAÇÃO DE PROCESSOS (LOGON INTERATIVO)

- No logon interativo o usuário, por intermédio de um terminal, fornece ao sistema um nome de identificação (username ou logon) e uma senha (password);
- O sistema operacional autentica estas informações verificando se estão corretamente cadastradas no arquivo de usuários;
- O arquivo de usuários é um arquivo do sistema operacional onde são armazenados todos os usuários autorizados a ter acesso ao sistema;
- Privilégios, quotas, permissões e o logoff.

# CRIAÇÃO DE PROCESSOS (LINGUAGEM DE COMANDO)

- Um usuário pode, a partir do seu processo, criar novos processos por intermédio de comandos da linguagem de comandos (shell do SO);
- Uma interface gráfica também pode ser liberada após a devida autenticação pelo Sistema Operacional, sendo o canal para criar novos processos;
- O processo criado pode ser foreground ou background, dependendo do comando de criação utilizado.



```
vi /home/fabio/arq1  
Ctrl+Z  
bg
```

```
fg
```

# criação de processos (rotina do so)

Um processo pode ser criado a partir de qualquer programa executável com o uso de rotinas do sistema operacional.

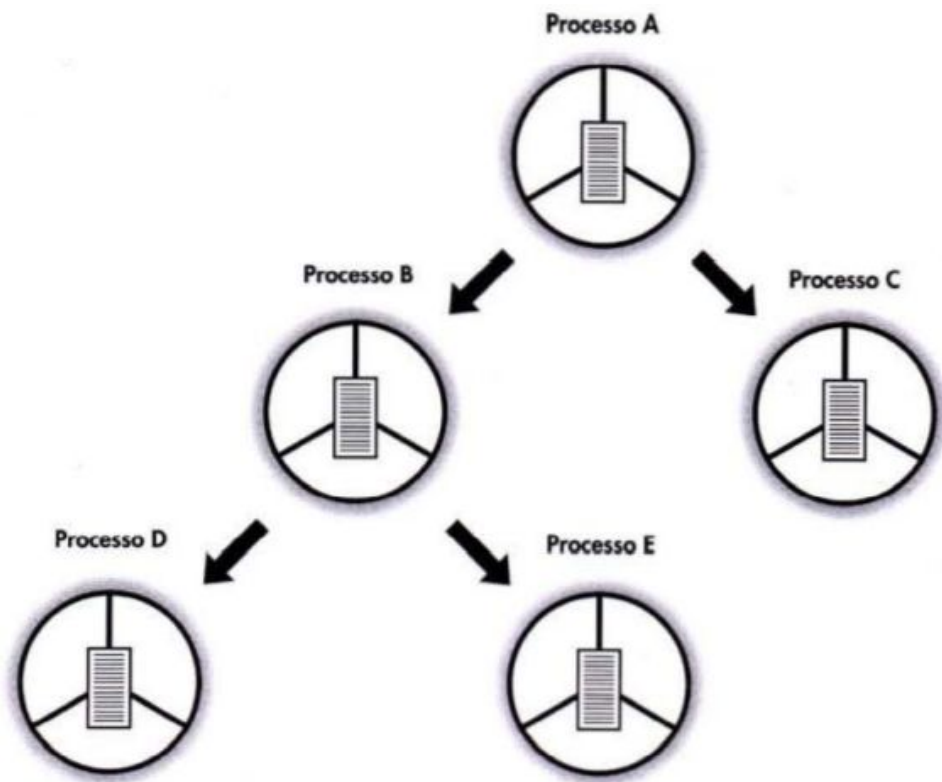
[illegible]

# PROCESSOS INDEPENDENTES

- Forma mais simples de implementar a concorrência em ambientes multiprogramáveis;
- Não existe a necessidade de vincular o processo criado ao seu criador;
- Apresenta uma estrutura completa de processo, tendo contexto de software, hardware e espaço de endereçamento;
- Nesse sentido, exige a alocação de um PCB.



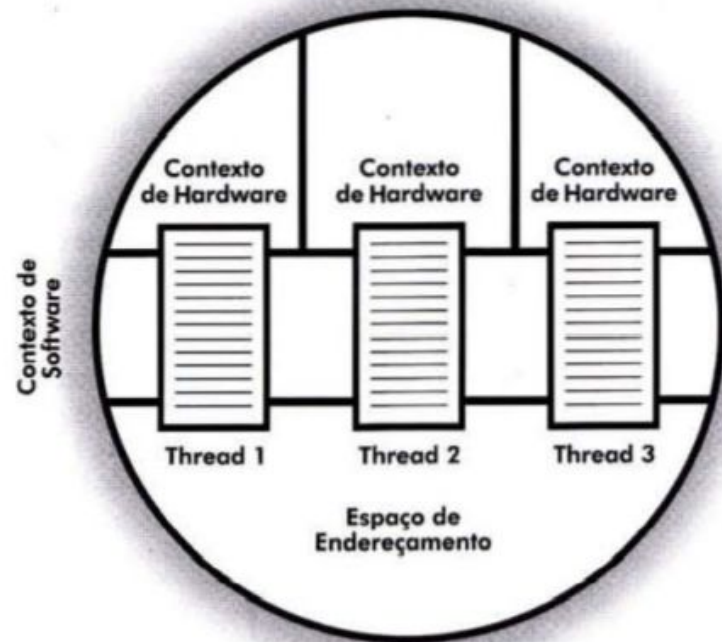
# SUBPROCESSOS



- São processos criados dentro de uma estrutura hierárquica;
- O processo criador é denominado processo-pai, enquanto o novo processo é chamado de subprocesso ou processo-filho;
- Dependência entre o processo criador e o subprocesso;
- Caso um processo-pai deixe de existir, os subprocessos subordinados são automaticamente eliminados;
- Os subprocessos possuem estrutura completa na PCB para identificá-los;
- Existe o compartilhamento de quotas com o processo-pai;

# THREADS

- Surgiu pela necessidade de otimizar os recursos exigidos pelo SO para criação de processos;
- Compartilha o Contexto de Software e Espaço de Endereçamento;
- Threads compartilham o processador da mesma maneira que um processo, ou seja, enquanto uma thread espera por uma operação de E/S, outra thread pode ser executada.



# PROCESSOS DO SISTEMA OPERACIONAL

- Estratégia utilizada para tirar o código do Kernel, deixando mais leve e estável;
- Alguns exemplos de serviços que os sistemas operacionais podem implementar como processos:
  - auditoria e segurança;
  - serviços de rede;
  - contabilização do uso de recursos;
  - contabilização de erros;
  - gerência de impressão;
  - gerência de jobs batch;
  - temporização;
  - comunicação de eventos;
  - interface de comandos (shell).

# SINAIS

- Mecanismo que permite notificar processos sobre eventos gerados pelo sistema operacional ou por outros processos;
- O uso de sinais é fundamental para a gerência de processos, além de possibilitar a comunicação e sincronização entre os mesmos;
- Observar a sinalização abaixo das teclas CTRL+C.



# SINAIS

- Podem ser utilizados em conjunto com temporizadores, no intuito de sinalizar ao processo algum evento associado ao tempo;
- Um exemplo seria a situação em que um processo deve ser avisado periodicamente para realizar alguma tarefa, como monitorar uma fila de pedidos;
- Depois de realizada a tarefa, o processo deve voltar a esperar pelo próximo sinal de temporização.

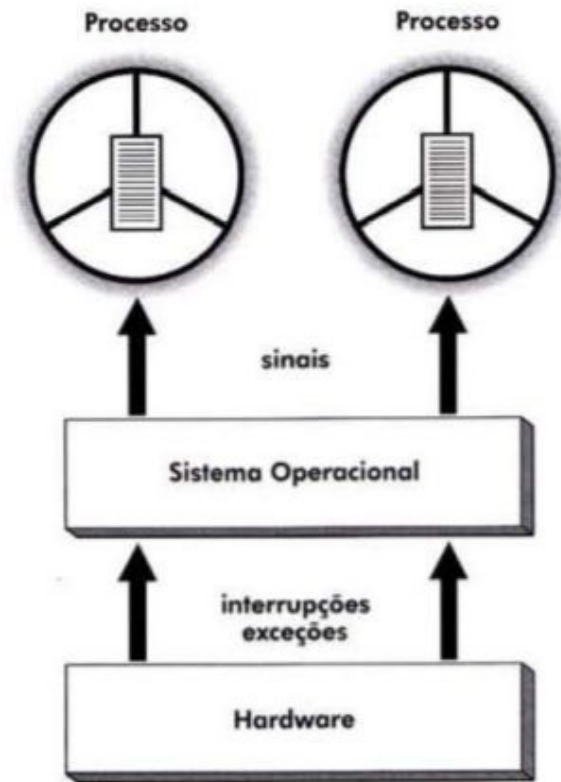
# SINAIS

- A maior parte dos eventos associados a sinais são gerados pelo sistema operacional ou pelo hardware, como a ocorrência de exceções, interrupções geradas por terminais, limites de quotas excedidos durante a execução dos processos e alarmes de tempo;
- Em outras situações, os eventos são gerados a partir de outros processos com o propósito de sincronizar suas execuções;
- **O sinal está para o processo assim como as interrupções e exceções estão para o sistema operacional.**



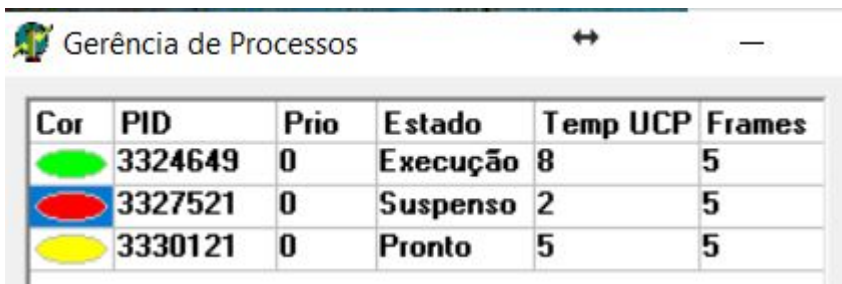
# SINAIS

- O tratamento de um sinal é muito semelhante ao mecanismo de interrupções. Quando um sinal é tratado, o contexto do processo é salvo e a execução desviada para um código de tratamento de sinal (signal handler), geralmente no núcleo do sistema;
- Após a execução do tratador de sinais, o programa pode voltar a ser processado do ponto onde foi interrompido.






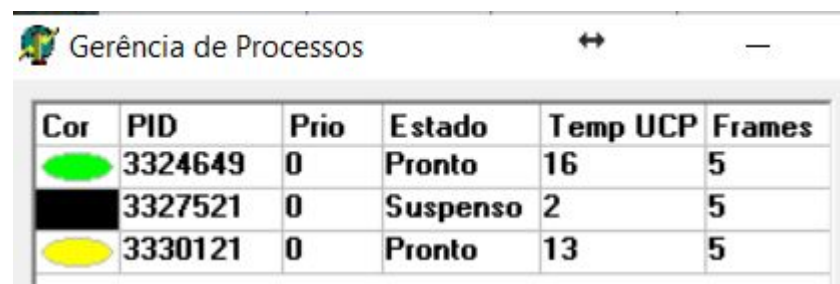
# SINAIS

- A geração de um sinal ocorre quando o sistema operacional, a partir da ocorrência de eventos síncronos ou assíncronos, notifica ao processo através de bits de sinalização localizados no seu PCB;
- Um processo não responde instantaneamente a um sinal. Os sinais ficam pendentes até que o processo seja escalonado, quando então serão tratados.






Gerência de Processos

Cor	PID	Prio	Estado	Temp UCP	Frames
	3324649	0	Execução	8	5
	3327521	0	Suspenso	2	5
	3330121	0	Pronto	5	5



Gerência de Processos

Cor	PID	Prio	Estado	Temp UCP	Frames
	3324649	0	Pronto	16	5
	3327521	0	Suspenso	2	5
	3330121	0	Pronto	13	5



# O QUE FICOU NA MENTE?

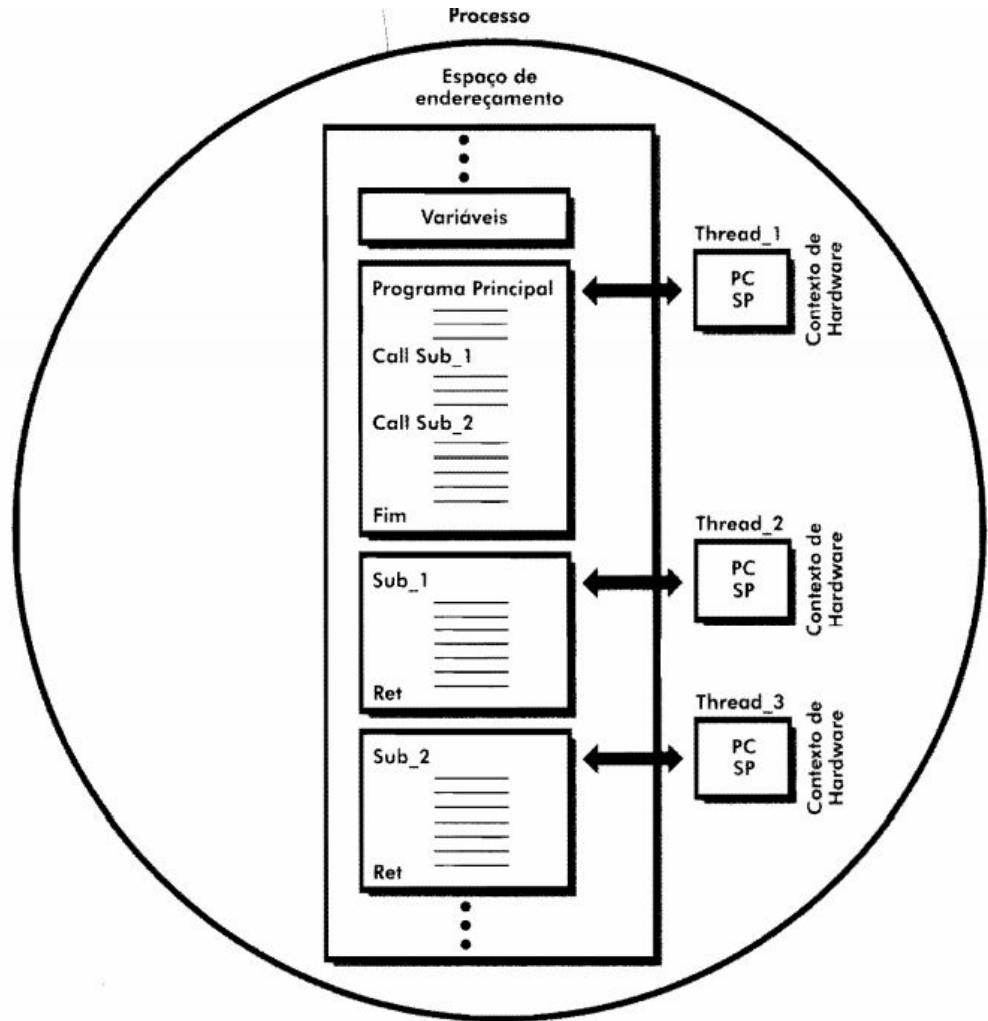
1. Defina os cinco estados possíveis de um processo.
2. Dê um exemplo que apresente todas as mudanças de estado de um processo, juntamente com o evento associado a cada mudança.
3. Diferencie processos multithreads e subprocessos.
4. Explique a diferença entre processos foreground e background.

# O QUE FICOU NA MENTE?

5. Qual a relação entre processo e arquitetura do Sistema Operacional que implementa serviços (microkernel)?
6. Dê exemplos de aplicações CPU-bound e I/O-bound.
7. Explique como a eliminação de um processo utiliza o mecanismo de sinais.
8. Quais as características de um processo independente?

# AMBIENTE MULTITHREAD

De forma simplificada, uma thread pode ser definida como uma sub-rotina de um programa executada de forma assíncrona, ou seja, executada paralelamente ao programa chamador.



# AMBIENTE MULTITHREAD

- O programador deve especificar as threads, associando-as às sub-rotinas assíncronas. Desta forma, um ambiente multithread possibilita a execução concorrente de sub-rotinas dentro de um mesmo processo;
- Cada processo pode responder a várias solicitações concorrentemente ou mesmo simultaneamente. caso haja mais de um processador;
- A grande vantagem do ambiente multithreads é a possibilidade de minimizar a alocação de recursos do sistema, além de diminuir o overhead na criação, troca e eliminação de processos.

# AMBIENTE MULTITHREAD

- Threads compartilham o processador da mesma maneira que processos e passam pelas mesmas mudanças de estado (execução. espera e pronto);
- Para permitir a troca de contexto entre as diversas threads, cada thread possui seu próprio contexto de hardware, com o conteúdo dos registradores gerais, PC e SP;
- **Dentro de um mesmo processo, threads compartilham o mesmo contexto de software e espaço de endereçamento com as demais threads, porém cada thread possui seu contexto de hardware individual.**

# AMBIENTE MULTITHREAD

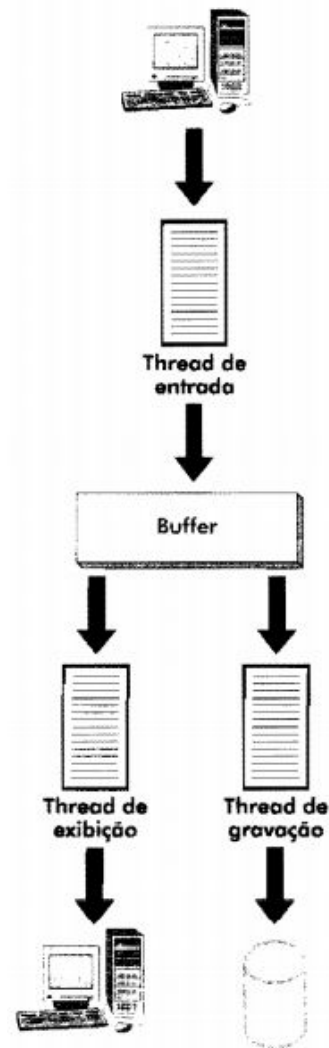
- Threads são implementados internamente através de uma estrutura de dados denominada bloco de controle do thread (Thread Control Block - TCB);
- O TCB armazena, além do contexto de hardware, mais algumas informações relacionadas exclusivamente a thread, como prioridade, estado de execução e bits de estado;
- Em um ambiente multithread, a unidade de alocação de recursos é o processo, onde todas as suas threads compartilham o espaço de endereçamento, descritores de arquivos e dispositivos de E/S;

# AMBIENTE MULTITHREAD

- O sistema operacional não seleciona um processo para a execução, mas sim uma de suas threads;
- A grande diferença entre aplicações monothread e multithread está no uso do espaço de endereçamento;
- Processos independentes e subprocessos possuem espaços de endereçamento individuais e protegidos, enquanto threads compartilham o espaço dentro de um mesmo processo;
- Esta característica permite que o compartilhamento de dados entre threads de um mesmo processo seja mais simples e rápido;

# AMBIENTE MULTITHREAD

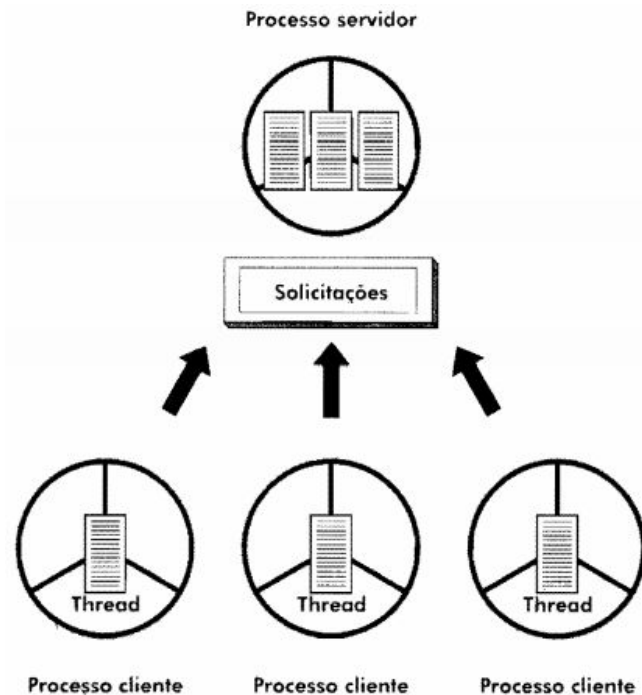
- Em algumas aplicações, a utilização de threads pode melhorar o desempenho apenas executando tarefas em background enquanto operações E/S estão sendo processadas;
- Editores de texto, planilhas, aplicativos gráficos e processadores de imagens são especialmente beneficiados quando desenvolvidos com base em threads.





# AMBIENTE MULTITHREAD E AMBIENTE CLIENTE/SERVIDOR

- Threads são essenciais para solicitações de serviços remotos;
- Permite solicitar o serviço remoto, enquanto a aplicação pode continuar realizando outras atividades;
- Em um ambiente monothread. se uma aplicação solicita um serviço remoto, ela pode ficar esperando indefinidamente. enquanto aguarda pelo resultado.



# O QUE FICOU NA MENTE?

1. Como uma aplicação pode implementar concorrência em um ambiente monothread?
2. Quais os problemas de aplicações concorrentes desenvolvidas em ambientes monothread?
3. O que é um ambiente multithread e quais as vantagens de sua utilização?
4. Qual estrutura de dados organiza as informações das threads criadas e o que ela armazena?