Resumo Sistema Operacional

Barbara Leticia da Silva CJ30299221

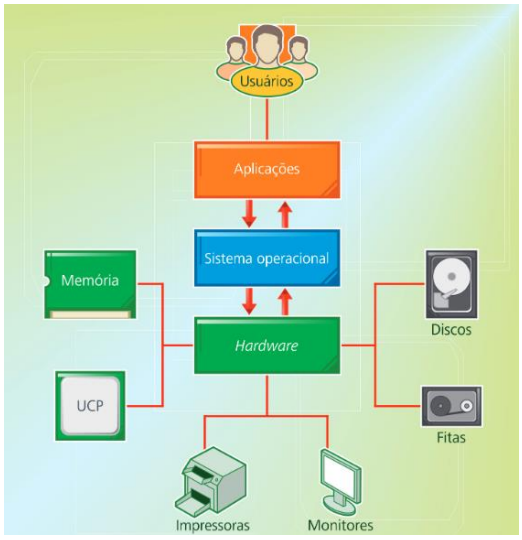
Marcos Vinicius de Souza Pereira CJ3025349

Murilo Ramos Froes de Paula CJ3017401

Raissa Pereira Miranda CJ3028941

Vitor Patrick dos Reis Revoredo CJ3029352

Sistema Operacional (S.O)



O sistema operacional se relaciona entre usuário, programa, hardware e sistema de arquivo, o principal programa existente sendo responsável por gerenciar e controlar todos os recursos do computador oferece a interface para o usuário se comunicar com o computador.

Os sistemas operacionais evoluíram ao longo do tempo, muito pela contribuição da evolução do hardware e das aplicações suportadas por ele. Os sistemas operacionais são divididos basicamente em três tipos.

Monoprogramável/Monotarefa

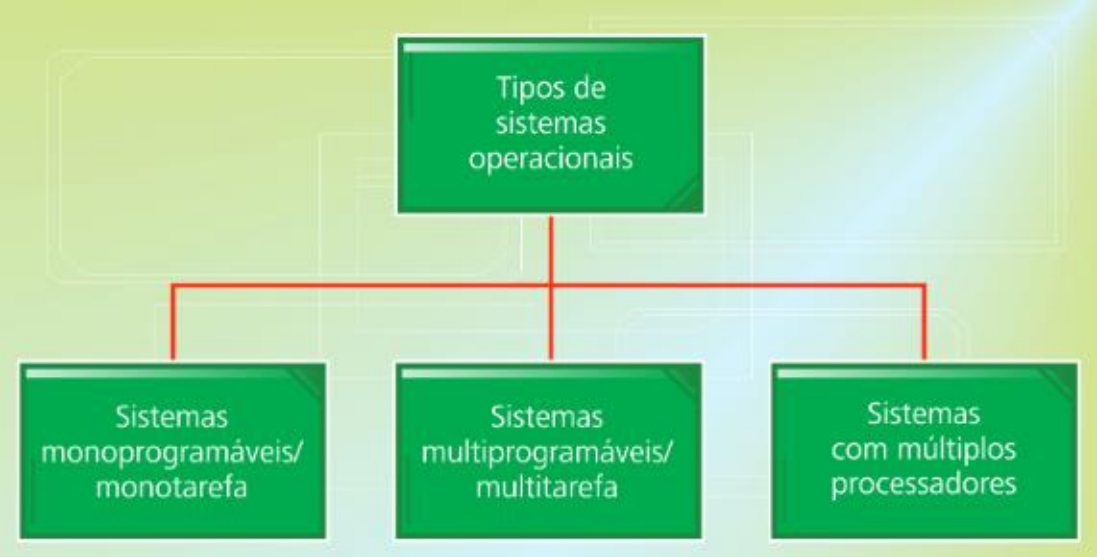
Se caracteriza por permitir que o processo a memória e os periféricos permanecem dedicados à execução de um **único processo** todos os recursos dedicam a uma única tarefa.

Multiprogramável/Multitarefa

Vários processos compartilham os recursos do computador, ou seja, o SO pode fazer mais de uma tarefa simultânea.

Multiprocessamento

Os sistemas de múltiplos processadores recebem este nome por possuírem dois ou mais processadores interligados trabalhando em conjunto.



Conforme podemos ver na imagem mostra os tipos de sistemas operacionais quanto a sua forma de operação. Estes dividem-se em: sistemas monoprogramáveis/monotarefa, sistemas multiprogramáveis/multitarefa e sistemas com múltiplos processadores. Dentre eles temos também o Monousuário e o Multiusuário:

Monousuário

Caracterizado por possuir um único usuário integrado.

Multiusuário

Permite que mais de um usuário acesse o computador.

Nota

É importante frisar que estas situações acima não são mutuamente exclusivas, ou seja:

1. Um programa **monousuário** pode ser **multitarefa**, pois pode ser que seja capaz de executar diversas tarefas em paralelo que atendam a pontualmente **um único usuário**. **Neste caso** teríamos um programa **monousuário** e **multitarefa**.

2. Um programa **multiusuário** pode ser **monotarefa**, pois apesar de permitir que seja acessado por **diversos usuários em simultâneo**, **executa** apenas **uma tarefa por vez**, ficando os **demais usuários aguardando sua vez na "fila"** para terem suas tarefas executadas pelo programa.

Mecanismo de Multiprogramação

Multiprogramação

Uma técnica que permite executar vários programas ao mesmo tempo no computador. Também é conhecida como multitarefa. O CPU alterna rapidamente entre os programas, permitindo a execução simultânea. Alguns do processo da multiprogramação:

Interrupção

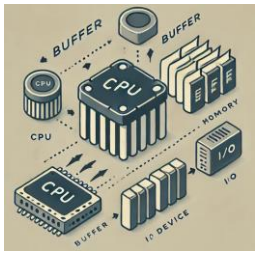
Obriga que o fluxo de execução seja alterado fazendo a interrupção para fazer outra coisa altera o processo do programa os dois tipos de introdução são software e hardware. Existem diversos tipos de interrupção alguns são

1. **Interrupções de Hardware** – Causadas por dispositivos externos, como teclado, mouse e discos rígidos.
2. **Interrupções de Software** – Geradas por programas para solicitar serviços do sistema operacional (ex: chamadas de sistema).
3. **Interrupções por Exceção** – Ocorrem devido a erros, como divisão por zero ou falha de memória.

Resumindo a Interrupções é fundamental para a eficiência dos sistemas operacionais modernos, permitindo resposta rápida a eventos e melhor gerenciamento de recursos.

Buffering

Uma técnica onde uma área de memória é utilizada nas operações de input (Entrada) ou o output (Saida), seu principal objetivo é **minimizar** a diferença de velocidade entre a CPU e os dispositivos periféricos.



A imagem ilustra o conceito de buffering em um sistema computacional, mostrando como os dados fluem entre a CPU, um buffer (memória temporária) e um **dispositivo de entrada/saída** (I/O device), como uma impressora.

Spooling

Semelhante ao buffer, mas salva no HD, ou seja, fica mais seguro os dados e são unidos e gravados em disco enquanto outros programas são processados.

Este arquivo em disco é denominado arquivo de SPOOL.

Simultaneuos

• Peripheral

• Operation

• On

• Line

Um exemplo do uso do spooling é:

A CPU gera dados e os armazena em um espaço especial do disco (spool).

O dispositivo de saída (ex: impressora) acessa os dados do spool quando estiver pronto para processá-los.

A CPU pode continuar executando outras tarefas enquanto o dispositivo de I/O lida com os dados.

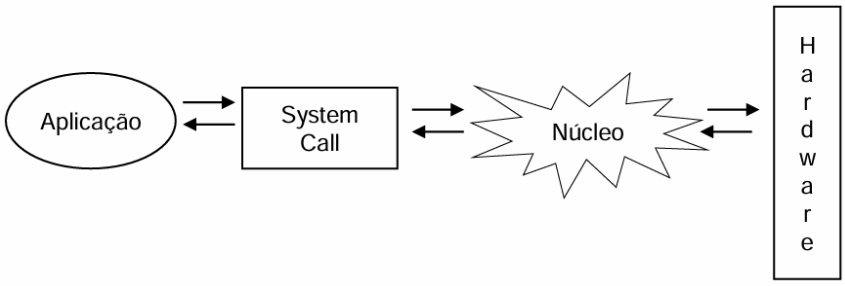
Resultando que dispositivos lentos funcionem sem atrasar a CPU ou outros processos.

Estruturas do Sistema Operacional

O sistema operacional é um projeto de software propriamente dito, suas definições básicas são programas de usuários que solicitam serviços ao sistema operacional através da execução de chamadas ao sistema está prática é chamada de system call. Através deste método existem diversas chamadas como por exemplo, para criar processos de gerenciar memoria, ler e gravar arquivos (I/O), gerenciar dispositivos de entrada/saída entre outros processos que o sistema possa necessitar.

Abaixo será exibido algumas imagens para ilustrar com o system call funciona:

System Calls



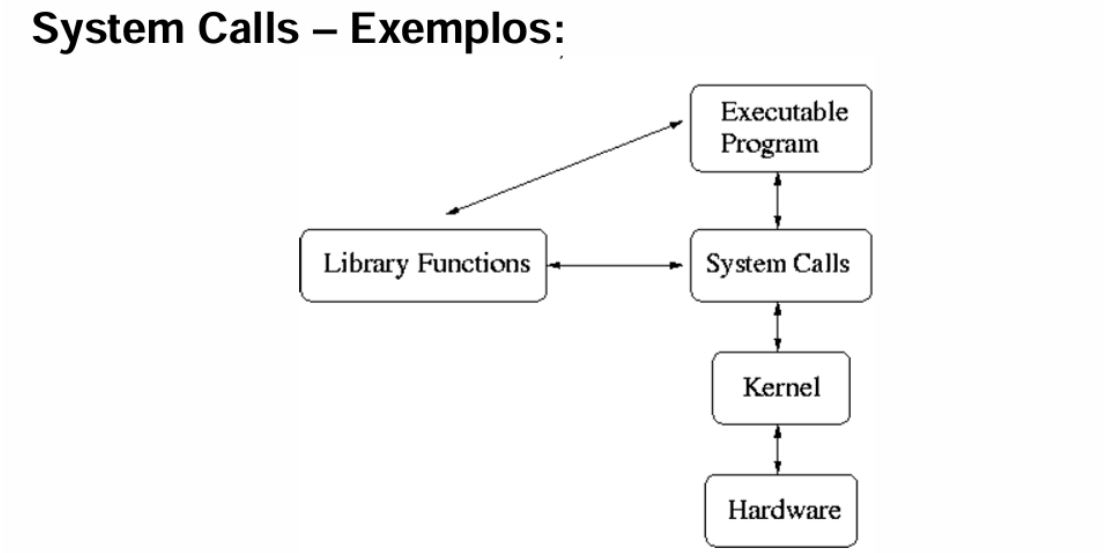
System Calls – Exemplos:

Windows	Unix	Descrição
CreateProcess	fork	Crie um novo processo
ExitProcess	exit	Termine a execução
CreateFile	read	Crie um arquivo/abra um existente
WriteFile	write	Escreva dados para um arquivo
CloseHandle	Close	Feche um arquivo
GetLocalTime	time	Obter horário atual
CreateDirectory	mkdir	Crie um novo diretório

System Calls – Exemplos:

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTime() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

System Calls – Exemplos:



Os modos de acesso do S.O são realizados por instruções das quais certas instruções podem ocasionar sérios danos a integridade do sistema como atualização de um arquivo/disco compartilhado, algumas operações podem ser feitas somente pelo SO como o I/O. Existem dois tipos de instruções as privilegiadas e as não-privilegiadas neste caso é necessário se atentar as instruções privilegiadas pois elas podem oferecer risco ao sistema, já as não-privilegiadas não apresentam perigo.

Para a execução a execução de uma instrução privilegiada há a implementação dos modos de acesso sendo eles.

Modo usuário: A aplicação só executa instruções não-privilegiadas acesso reduzido de instruções.

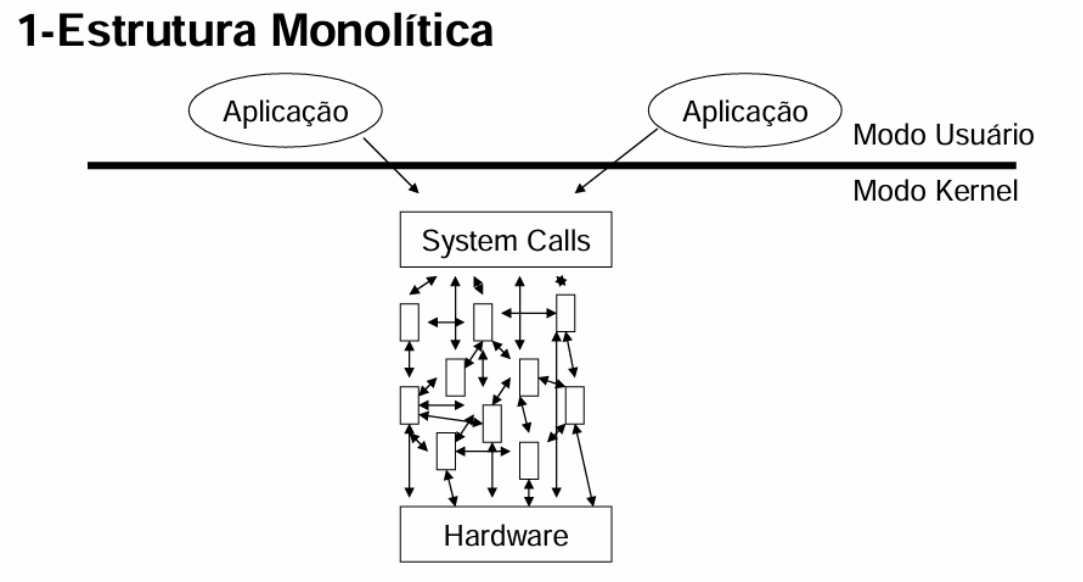
Modo kernel: A aplicação tem acesso total ao conjunto de instruções.

Dentro da CPU existe um registrador especial chamado PSW (program status word) que gerencia se a instrução pode ou não, ser executada pela aplicação. O núcleo do sistema sempre roda em modo kernel gerenciando e compartilhando todos os recursos, o procedimento de execução é a requisição da operação (system call), a chamada muda para modo kernel, é feito o processamento e retorna para o usuário.

As estruturas do sistema operacional são sistema monolíticos, estruturas em camadas, microkernel, modelo cliente-servidor e máquinas virtuais.

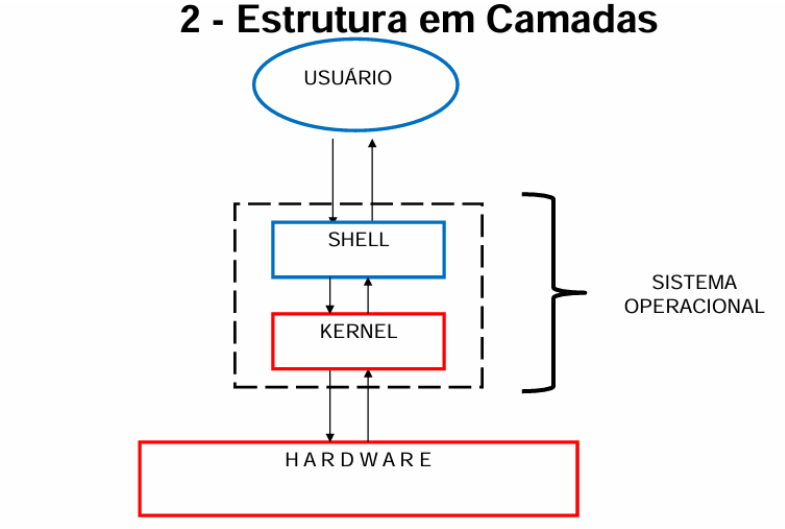
Estrutura monolítica: neste sistema não existem separação entre as rotinas que atendem ao usuário e as rotinas que gerenciam o funcionamento do hardware, todas estão no mesmo nível

do software sendo assim esta situação não seria a ideal pois as rotinas que comandam o hardware podem ser facilmente acessadas pelo usuário podem causar instabilidade no sistema.



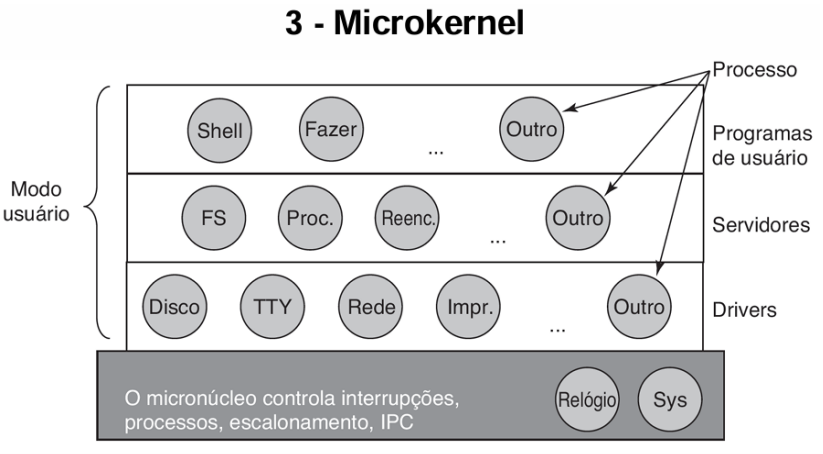
Estruturas em camadas: os sistemas operacionais são organizados em camadas para simplificar a complexidade dos comandos apesar de parecerem simples são na realidade muitos complicados, desta forma é mais interessante separar as rotinas que servem para a interface do usuário, das rotinas que tratam do funcionamento do hardware. Um é exemplo a considerar é um comando que solicita a impressão de um arquivo e é exigido diversas partes na estrutura como determinar se o arquivo existe, determinar se o usuário tem permissão para acessar o arquivo, localizar o arquivo, ler o arquivo, determinar onde o arquivo vai ser impresso, imprimir no dispositivo de saída.

Muitos sistemas operacionais implementam a interface de camadas entre o usuário e o computador, esta abordagem é conhecida com top down (de cima para baixo). A camada superior (shell) define as funções e a inferior (Kernel) contém a parte de baixo nível para executar. A figura a seguir demonstrara como a camada superior comunica com o usuário através de interface chamada de interpretador de comandos (shell) esta parte é a que usuário está mais familiarizado, a última camada é o núcleo (kernel) podendo ser interpretado como o cérebro do sistema, sendo a parte de baixo nível onde vão os códigos que controlam o hardware para o gerenciamento de recursos para todos os programas.

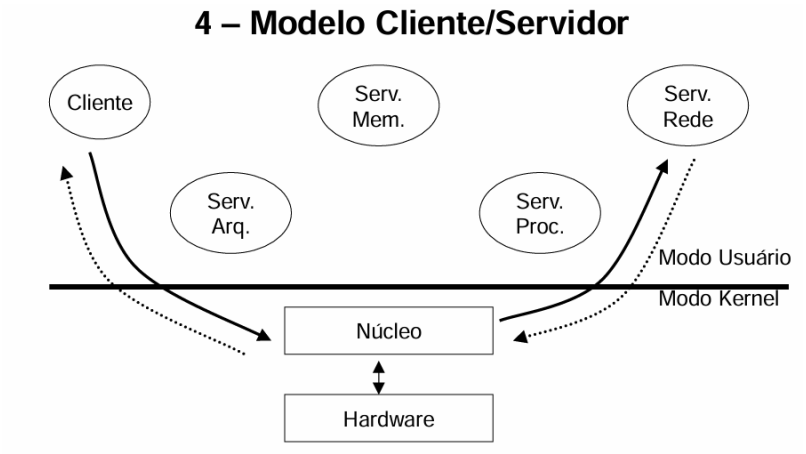


As principais funções do kernel são tratamento de interrupções, criação e eliminação de processos, sincronização e comunicação entre processos, escalonamento e controle de processos, gerencia de memória, gerencia de sistema de arquivos, operação de entrada e saída, segurança do sistema.

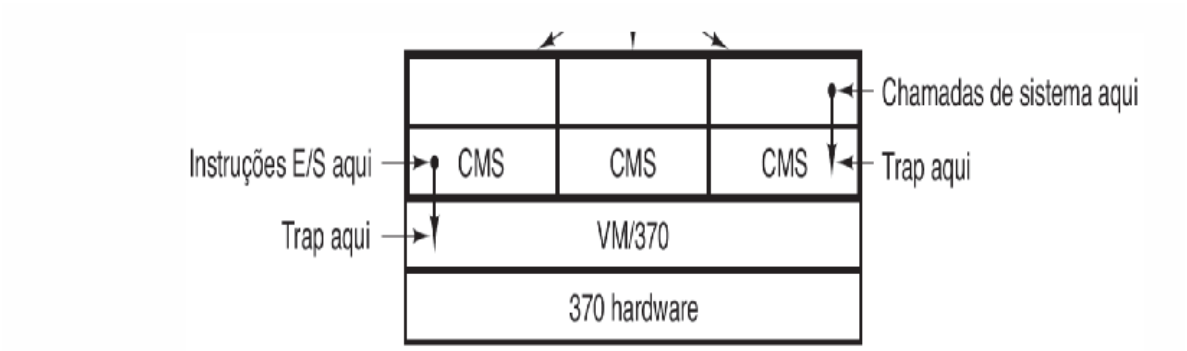
MicroKernel: Nos primórdios todas as camadas entravam no núcleo, mas conforme citamos no sistema anterior existem bugs (erros) que devem ser evitados colocando funcionalidades do núcleo no espaço do usuário. A principal ideia é se obter alta confiabilidade para isso o sistema operacional é dividido em pequenos módulos bem definidos destes apenas um modulo (micronúcleo) sendo executado em modo kernel. Suas aliciações são em indústria, aeronáutica e militar, alguns exemplos de sistemas são o symbian e minix. Os benefícios se devem a facilidade de estender um microkernel, facilidade de dimensionar o SO para novas arquiteturas e sua confiabilidade se da por ter menos códigos sendo executado em modo kernel.



Modelo Cliente/Servidor: variando da ideia de microkernel, sua definição é duas classes de processos servidores e clientes, os servidores prestam o serviço e os clientes os consomem. A generalização desta ideia e a capacidade de executar clientes e servidores em diversos computadores diferentes conectados através de uma rede local, a comunicação entre cliente servidor e feito por troca de mensagens.



Máquinas Virtuais: ao contrário dos sistemas anteriores as máquinas não são entendidas, trabalham exatamente com um hardware podendo cada uma executar um sistema operacional. O modelo z/VM da IBM (serie z), é capaz de executar multiplos sistemas como linux ao lado de sistemas tradicionais da IBM.



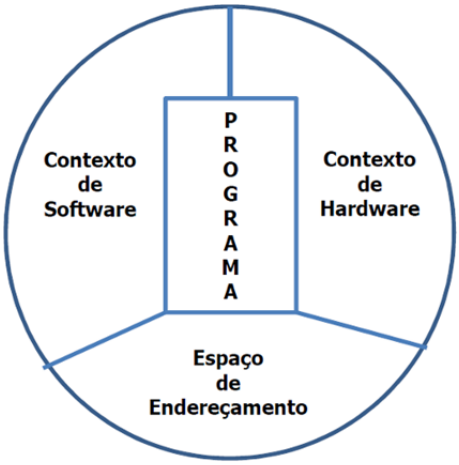
O conceito de maquia virtual parece ser recente, mas a IBM já havia adotado este produto a quatro décadas, mas como nos últimos anos foram se criando necessidades, novos softwares e novas tecnologias, a ideia se tornou um tópico interessante. Muitas empresas trabalhavam com diversos servidores (correia, servidores web, ftp...) em máquinas separadas, muitos também com sistemas operacionais diferentes, virtualização foi a chave encontrada para se executar todos eles na mesma máquina. Empresa de hospedagem observaram a oportunidade

de lucrar já que a virtualização permite que possa ser hospedado vários clientes em uma única máquina fisica.

Introdução aos Processos

O **processo**, também chamado de tarefa (task), é um programa em execução, formado pelo código executável, pelos dados referentes ao código e outras informações para a execução do programa.

Um processo pode ser representado como:



Contexto de Software: características do processo incluídas na execução de um programa, divididas em:

> **Identificação:** número de identificação do processo (PID) ou identificação do usuário (UID) que o criou;
>
> **Quotas:** limites de cada recurso do sistema que um processo pode alocar;
>
> **Privilégios:** o que o processo pode ou não fazer em relação ao sistema e aos outros processos.

Contexto de Hardware: formado pelo conteúdo dos registradores. Quando ocorre a troca de processos na CPU, pelo SO, chamamos isso de mudança de contexto. Essa mudança salva o conteúdo dos registradores na CPU e carrega eles com os valores do processo que está ganhando a utilização do processador.

Espaço de Endereçamento: área da memória do processo em que o programa será executado e para dados utilizados por ele. Devendo ser protegido do espaço de outros processos.

PCB - Process Control Block / Blocos de Controle de Processo

O sistema operacional possui uma lista de blocos de controle de processo (PCB), onde estão as informações que o SO precisa saber sobre o processo, como:

- **Número de identificação de processo**: PID;
- **Estado do processo:** o SO atualiza seu PCB assim que o processo passa pelas transições de estado;
- **Tempo máximo de execução e o tempo acumulado de execução:** o sistema controla quanto tempo da CPU um processo em execução usa. Com isso, o tempo de execução acumulado não pode passar o tempo máximo permitido;
- **Recursos e limites atuais:** o processo em execução pode requerer mais memória ou espaço em disco, sendo o SO que controla o que será autorizado a cada processo;
- **Prioridade de processo:** o sistema pode escalonar um processo, ou proporcionar a ele o acesso aos recursos, com base na sua prioridade.
- **Áreas de armazenamento:** se um processo é interrompido temporariamente, o sistema tem de salvar os valores dos registradores. Após isso, o sistema pode restaurar esses valores e o processo pode retomar a execução no ponto exato em que parou.

Cada processo possui um PCB. Quando um processo é iniciado, o SO cria um PCB para ele e o mantém na lista de PCB's. Ao terminar o processo, o sistema remove o seu PCB da lista.

Estados do Processo

1. **Pronto:** é quando um processo (ou vários no sistema) está temporariamente parado enquanto um outro é executado.
    - fila de processos PRONTOS é mantida ordenada pelo algoritmo de escalonamento.
2. **Executando:** é quando um processo está sendo executado pela CPU, revezando com outros processos conforme uma regra estabelecida pelo SO.
3. **Bloqueado:** é quando um processo está não-executável até que um evento externo aconteça.

- A fila de processos BLOQUEADOS é mantida desordenada visto que os eventos ocorrem de forma imprevisível.

Considerando-se o caso de uma única CPU, somente um processo pode estar EXECUTANDO enquanto vários processos podem estar em uma fila de PRONTOS ou BLOQUEADOS.

1. Programa é chamado, um processo então é criado e inserido no final da fila de PRONTOS.
2. O processo aos poucos vai chegando ao início da mesma.
3. Quando o processo fica em primeiro na fila e a CPU torna-se disponível, o processo recebe a CPU para execução (transição do estado de PRONTO para EXECUTANDO).
4. Se um processo que está EXECUTANDO requisita uma operação de entrada/saída ele voluntariamente libera a CPU entregando o controle para o SO.
5. Este por sua vez insere o processo atual na fila de BLOQUEADOS e entrega a CPU para o próximo processo da fila de PRONTOS.
6. Quando o término da operação de entrada/saída ocorrer, o SO é sinalizado e o processo retirado da fila de BLOQUEADOS e inserido no final da fila de PRONTOS.

Transições de Estados do Processo

Pronto à Execução: ocorre quando a CPU está livre e o primeiro processo da fila de prontos recebe o recurso CPU.

Execução à Bloqueado: ocorre quando o processo que está executando, necessita de algum evento externo.

Executando à Pronto: ocorre em algoritmos preemptivos.

Bloqueado à Pronto: ocorre quando o evento externo é liberado e o processo retorna para a fila de prontos.

Pronto à Bloqueado e **Bloqueado à Executando** **NÃO EXISTEM!**

Um processo bloqueado deve EXECUTAR algum evento externo, então não tem como um processo que está pronto ir direto. Assim como um processo bloqueado deve sempre retornar à fila de prontos.

Processos CPU-Bound e I/O-Bound

Processos CPU-Bound: ficam maior parte do tempo no estado de execução e pronto, precisando de muito tempo de CPU.

Processos I/O-Bound: ficam maior parte do tempo no estado de bloqueado, precisando de muitas operações de I/O.

Escalonamento de Processos e seus Principais Algoritmos

Definição de Escalonamento de Processos: O escalonamento de processos é um processo de escolha e distribuição de recursos do sistema para os processos que estão em execução. O uso da CPU é fundamental para assegurar o desempenho do Sistema Operacional.

Importância do Escalonamento: O escalonamento é essencial para o bom funcionamento do sistema, é ele de define como os processos são supridos e coordenados.

Algoritmo FCFS(*First-Come,First-Served*): Os processos são executados por ordem de chegada no sistema, desconsiderando seu período de execução.

Algoritmo SJF(*Shortest Job First*): O SJF dá preferência ao processo com o menor tempo de execução, reduzindo o tempo médio de espera.

Algoritmo de Prioridade: O processo com a maior prioridade será atendido primeiro. Cada um dos processos é designado a sua respectiva prioridade.

Algoritmo Round Robin: Cada um dos processos irá adquirir uma parte de tempo na CPU, que se chama quantum, antes que seja preemptado.

Recapitulando os conceitos da ordem dos processos com cada Algoritmo: Os Processos de um a seis serão denominados como: **P1, P2, P3, P4, P5 e P6.**

Processo	Tempo de Execução	Prioridade	Quantum
P1	10	Alta	2
P2	5	Baixa	3
P3	8	Média	4
P4	4	Alta	2
P5	6	Baixa	3
P6	7	Média	4

Tipos de escalonamento de processos

Escalonamento Preemptivo

Esse tipo de escalonamento permite com que o sistema operacional interrompa um processo que está sendo executado para dar lugar a outro, desse modo isso garante com que o sistema consiga dar resposta rápida a processos com maior tempo de CPU.

O sistema apresenta certas características sendo que um processo em execução pode ser interrompido e outro pode ser executado no lugar, como também recebem um tempo limitado para a execução, contudo em um escalonamento preemptivo ele garante que todos os processos recebam tempo de CPU, de tal forma que podem sobrecarregar o sistema devido as trocas de execução de processos.

Escalonamento Não Preemptivo

Enquanto no escalonamento preemptivo, o processo quando iniciado para a execução pode ser interrompido para a execução de outro, em um escalonamento não preemptivo, o processo não será interrompido até que termine ou entre em estado de espera.

Escalonamento por Prioridade

É caracterizado por antes do processo ser executado, ele recebe um nível de prioridade, assim o processo com maior prioridade é escolhido para a execução, esse tipo de escalonamento pode ser Preemptivo ou Não Preemptivo, variando de como o sistema lida com a chegada dos processos de alta prioridade, os que possuem maior prioridade são os primeiros a serem executados, em certas ocasiões processos de baixa prioridade podem nunca ser executados se sempre houver processos de maior prioridade chegando, esse tipo de situação é chamada de **Starvation**.

Escalonamento por Tempo Real

É configurado como um sistema que é utilizado em sistemas do qual exigem rigorosas garantias de tempo, como sistemas embarcados, controle de processos industriais e sistemas de comunicação em tempo real, nesse tipo de escalonamento os processos deverão ser executados dentro de prazos específicos.

Algoritmos de Escalonamento de Processos

Algoritmo (FIFO) (First In First Out)

Basicamente sendo um dos algoritmos mais básicos e fáceis de serem aplicáveis, seu funcionamento ocorre de acordo com a ordem de chegada dos processos, assim o primeiro processo a chegar para ser executado será o primeiro a ser finalizado, vale ressaltar que sendo um algoritmo de caraterística Não Preemptivo, um processo quando iniciado não pode ser interrompido até que termine a execução.

Algoritmo (SJF) (Shorthest Job First)

Esse algoritmo tem como objetivo minimizar o tempo médio de espera dos processos, sendo que é escolhido os processos com menor tempo de execução a serem executados primeiro, a

execução dos processos poderá ocorrer de duas maneiras sendo a primeira de maneira Preemptiva e outra sendo Não Preemptiva, ocorre de método Preemptivo se um processo chegar com tempo de duração da CPU menor que a de um processo em execução, já um método Não Preemptivo ocorre se um processo ganha espaço na CPU ele é executado até o fim do tempo de sua utilização.

Algoritmo (RR) (Round Robin)

Esse algoritmo é vinculado como sendo um dos mais utilizados em sistemas operacionais, ainda mais se tratando de sistemas multitarefa, basicamente todos os processos a serem executados recebem valores aproximados de tempo de utilização da CPU, sendo um algoritmo que opera com base no método Preemptivo, equilibrando o tempo de processamento entre os processos, se os tempos de execução dos processos forem bastante distintos um do outro, os processos com duração menor terão seu tempo de resposta afetado, oferecendo margem para processos com duração maior ocuparem esse espaço dos processos com duração menor.

Questões de Fixação

1. Qual algoritmo prioriza o menor tempo de execução?
   R: **B) SJF.**

2. Qual algoritmo atende os processos por ordem de chegada?
   R: **C) FCFS.**

3. Objetivo do escalonamento de processos?
   R: **A) Maximizar a utilização da CPU.**

4. Principal desvantagem do Round Robin com quantum pequeno?
   R: **B) Tempo de espera prolongado para processos longos.**

5. Algoritmo com tempo pré-definido antes da preempção?
   R: **C) Round Robin.**

Tipos de Sistemas Operacionais

1. Batch / Lote

- Processam tarefas em lotes, **sem interação do usuário durante o processo**.
- **Exemplo:** Folha de pagamento processada à noite em uma empresa.

2. Interativo

- Permitem **interação direta do usuário** por meio de interfaces gráficas ou linha de comando.
- **Exemplo:** Windows, MacOS, Linux (GUI ou Shell).

3. Tempo Real

- Respondem a eventos **em tempo preciso**, sendo usados em sistemas críticos.
- **Exemplo:** Controle de tráfego aéreo, monitoramento médico.

Questões de Fixação

1. Sistema que processa tarefas em lotes, sem interação?
   R: **A) Batch/Lote.**

2. Sistema que permite interação por GUI ou linha de comando?
   R: **B) Interativo.**

3. Sistema que responde a eventos em tempo preciso?
   R: **C) Tempo Real.**

Terminal Linux

O que é?

O **Terminal Linux** é uma interface de linha de comando (CLI) que permite interação direta com o sistema, ideal para automação e controle.

Como abrir?

- **Ubuntu/Debian:** Ctrl + Alt + T.
- **Fedora:** Buscar por "Terminal".
- **MacOS:** Acessar via Finder.

Comandos de Navegação

- pwd: Caminho atual.
- ls: Lista arquivos e pastas.
- cd <diretório>: Acessar diretório.
- cd .. : Voltar um nível.
- cd /: Ir à raiz.
- cd ~: Ir à pasta do usuário.

Criar/Excluir Arquivos e Pastas

- touch <nome>: Criar arquivo.
- mkdir <nome>: Criar pasta.
- rm <arquivo>: Remover arquivo.
- rm -r <pasta>: Remover pasta e conteúdo.

Exibir e Editar Arquivos

- cat: Exibir conteúdo.
- less: Ler por páginas.
- nano: Editar (Ctrl+X, Y, Enter).

- head -n 5: Ver primeiras 5 linhas.
- tail -n 5: Ver últimas 5 linhas.

Copiar, Mover e Renomear

- cp: Copiar arquivo.
- cp -r: Copiar pasta.
- mv: Mover ou renomear.

Permissões e Usuários

- ls -l: Ver permissões.
- chmod 755: Alterar permissões.
- chown: Alterar dono (usar sudo se necessário).

Gerenciar Processos

- ps aux: Listar processos.
- top: Ver processos em tempo real.
- htop: Versão avançada (instalar com sudo apt install htop).
- kill <PID>: Finalizar processo.
- killall <programa>: Finalizar todos os processos do programa.

Administração e Atualização

- sudo <comando>: Executar como admin.
- sudo apt update && sudo apt upgrade -y: Atualizar pacotes.

Dicas Finais

- Principais comandos:
    - **Navegação:** pwd, ls, cd.
    - **Manipulação de arquivos:** touch, mkdir, rm, cp, mv.
    - **Permissões:** chmod, chown.
    - **Processos:** ps, top, kill.

    - **Administração:** sudo, apt update.
- Use man <comando> para ver o manual do comando.
    - Exemplo: man ls