PRCMP - Teoria

T03- sistemas convecnionais

Arquitetura de von Neumann tem quatro sistemas:

Esta arquitetura pode guardar número binários, as instruções tem de estar codificadas em números, o processador da um conjunto de instruções em que são mapeadas todas as instruções e o seu respetivo ID. Para cada instrução o CU define que circuitos devem operar.

A diagram of a system

Description automatically generated

* Memoria: guarda as instruções e informação.

RAM( Random Access Memory): so guarda informacao enquando o computador esta ligado

ROM (Reas only Memory) guarda tarefas que nunca mudam como iniciar o PC.

A memoria esta organizada em grupos de 8 bits sendo que cada celulu de memoria pode ser acedida pelo seu ID

* I/O: comunica com o utilizador: ecrã, teclado, disco

Guarda informacao em massa os seus tipos de memoria são: Direct Access Storage Devices (DASD): e tem pens discos rígidos etc e Sequential Access Storage Devices (SASD)

São lentos a comparar com a memoria, cada dispositivo tem um controlador que executa autonomamente operacoes pedidas pelo processador.

Direct Memory Access (DMA): informacao é transferida entre a memoria e o controlador do I/O

* ALU: Executar cálculos aritméticos e lógicos que o programa precisa
* Registos: guarda a informação usada pelo o CU e ALU
* CU: controla o processamente de informações e instruções dentro do CPU.

Utiliza o ciclo: Fetch (vai buscar da memoria as instreucoes a executar)🡪 decode( descodifica a informacao e determina o que fazer) 🡪 execute

System Bus: faz com que todos estes componentes possam comunicar entre si

Técnicas que podem melhorar a performace do CPU:

Instruction pipelining

Existem vários ciclos e durante cada etapa do ciclo de fetch decode execute partes de um processo são feitas em simultâneo. Estas etapas são feitas em diferentes circuitos permitindo que a execução das instruções seja menor.

A chart with green and black text

Description automatically generated

Processador superescalar

Esta técnica utiliza muitas unidades de execução, mas só uma unidade de controlo. As unidades de execução contem:

* ALU para cálculos de integers
* Floating point units FPU para cálculos com casas decimais
* Unidades de memoria
* Branch units para fazer condições.

Esta técnica tira partido instruction-level parallelism (ILP) em que instruções são executa em paralelo e não sequencialmente.

Scheduling dinâmico: o processado de forma dinâmica decide quais instruções vão ser executadas em paralelo baseado nas dependências, disponibilidade das unidades de execução entre outros fatores

Reordenação de instruções: o processador acompanha as dependências entre instruções e reordena para maximizar o paralelismo

Previsão da ramificação: serve para minimizar os delays entre as instruções para manter o flow das decisões

Memoria cache

Memoria de alta velocidade localizada no mesmo chip do CPU, reduz o teu que o CPU precisa para aceder a informação se tivesse de ir a RAM.

Cache miss acorre quando foi acedida, mas a informação não esta la e por isso o CPU precisa de ir a RAM.

Multicores Chips: integração de vários chips que não precisam de muita energia no mesmo CPU. A maior quantidade de processamento acontece, pois, o paralelismo foi facilitado, pois existem programas a serem executado ao mesmo tempo, ou ate a correr vários segmentos do mesmo programa ao mesmo tempo isto chama-se (multithreading)

GPU: são focados na parte gráfica dos computadores sendo que tem instruções otimizadas parta a parte gráfica como renderizar gráficos e computações paralelas.

Computações paralelas: são a realização da mesmas instruções através de varias data streams ao mesmo tempo

Papel geral: a função do GPU ao contrário do CPU que é fazer computações gerais, o GPU faz computações especializadas para melhorar a performance geral do sistema todo.

T04- Software

**Hardware*: parte física e difícil de modificar permite implementar software.***

**Software**: conjunto de instruções que são guardadas e executadas pelo hardware este é fácil de mudar.

Instruções para o processador são passadas pela instruction set architecture (ISA), esta define a interface entre o software e o hardware e especifica o modelo abstrato do computador:

* Modelo de execução
* O endereço e os formatos de informação
* O registo de processos e as instruções

Para o processador implementar a ISA precisa de utilizar logica que permite que instruções sejam executadas. O design logico, informação e caminhos presentes num microprocessador chama-se microarquitetura.

Instruções da máquina: são codificadas em grupos de bytes. Opcode indica a operação a ser executada, indicando o endereço ou a informação a ser utilizada.

Linguagens de alto nível: tem como vantagens serem mais fácil para os humanos escreverem, dão abstração que não depende na máquina, apesar disso é preciso traduzir o código fonte para código de máquina

Programas interpretados: um interpretador é um programa que executa instruções de um script sem produzir um executável.

Programas compilados: compilador é um programa que traduz código de um programa inteiro para código que a máquina entende, este código pode ser carregado executado diretamente pelo processador depois de compilado.

Passos de um programa compilado:

Construir um executável que traduz uma linguagem de alto nível para código da máquina. É preciso fundir código de diferentes módulos e libraries para obter um ficheiro executável.

1. Compilação: compila o código source em linguagem de montagem que pode ser lida por humanos e interpretada para encontrar o objetivo da ISA. Esta parte do código mantem os nomes das variáveis, apesar de traduzir para instruções de máquina
2. Juntar o código: o assembeler converte o código para binário e gere um ficheiro como objeto, todos os valores e símbolos são substituídos pelos seus endereços
3. Ligação entre as diferentes partes: o linker junta todas as partes do código (os módulos) num só modulo, se estiver a usar libraries o linkar vai liga-la com o modulo compilado, transformando-o num executável.

Compilação cruzada: compilação de um programa para uma arquitetura diferente, pode ser uma arquitetura com restrições em recursos, baixa performance ou arquitetura minimalista.

Ferramentas de compilação incluem a library da arquitetura de destino, para que o programa seja construído com mais recursos e poderoso com for aplicado.

Layout de memoria de um programa: um programa requer instruções (secção de texto) e informação (secção de informação).

A screen shot of a computer

Description automatically generated

**Secção de texto:** contem as instruções a executar, tem tamanho fixo e só pode ser lido para evitar mudanças localizado no ROM ou na RAM de for carregado de um ficheiro.

**Secção de texto:** esta localizada na RAM

* Segmento de informação: variáveis globais e estáticas são determinadas e compiladas ficando persistentes durante a execução do programa, este segmento tem um tamanho fixo durante a execução do programa.
* Stack: guarda parâmetros de funções, variáveis locais e endereços de retorno, expande e diminui com a chamada de funções.
* Heap: permite memoria dinâmica e alocação durante o tempo em que o programa corre, sendo útil para estruturas de dados, arrays e instalação de objetos.

T05- Introdução a sistemas operativos

Propósito de um computador: executa programas para resolver problema dados por um utilizador, utilizar um programa num computador é o normal e é o mais eficiente e funciona.

Apesar disso os computadores usam sistemas operativos, que vem com custos adicionais de espaço, tempo adicional que usa poder de processamento.

**Processamento em serie**: no início os programas não tinham sistema operativo e corriam um programa de cada vez, sendo que esta programas eram carregados manualmente na memoria e inicializados manualmente numa consola, não corriam com input de um utilizador e utilizavam informação da execução anterior.

Colocar um computador a executar um programa demorava muito tempo pois eram os humanos a fazer tudo, tal como corrigir erros pois era preciso verificar a memoria o código da máquina e os registos físicos.

Assim os computadores eram caros, mas a sua utilização era baixa.

**Processamento em Batch:** o hardware foi aumentando o tempo de resposta apesar de continuar não interativo, linguagens de alto nível e libraries como funções comuns simplificaram a programação, cada programa precisava de compilação e carregar antes de ser executado adicionando mais complexidade e reduzindo a eficiência.

A solução foi automatizar com a criação de jobs, por isso foi criado o primeiro sistema operativo o residente monitor, o monitor permaneceu residente na memoria durante toda a operação do computador quando termina carrega o programa seguinte.

Para controlar a sequência de programas o operador poe cartões com os programas e define a sequencia, sendo que, existe um interpretador de cartões que lê e executa as instruções nos cartões e carrega o programa na memoria esta sequência repete automaticamente até todos os programas terem sido carregados.

Este tipo de processamento continua a ser relevante pois permite aos computadores gerir grande volume de tarefas repetitivas como filtrar informação.

Avanços na tecnologia: operações de I/O automáticas, mais memoria, inovação nos disco para permitir guardar mais informações, interação com o utilizador aprimorada programas podem ser carregados do disco e salvo diretamente no mesmo.

MultiProgramming: tem características como-

* Carregamento de programas em simultâneo: vários programas carregados na memoria principal
* Execução intervalada: o CPU intervala a execução de programas na RAM.
* Utilização de recursos eficiente
* Otimização da utilização do CPU

**Tim-Sharing:** programas interativos que esperam pelo input de utilizadores em quando o CPU executa outro programas. Muito utilizadores utilizam os mesmos recursos fazendo um uso mais eficiente do computador. O utilizar interage com o sistema usando o terminal que esta ligado ao computador por linha de comandos ou uma rede.

Sistemas dedicados em correr um único programa não precisam de sistema operativo

Sistema a correr vários programas ao mesmo tempo precisam de sistema operativo para gerir os recursos do computador.

Assim os sistemas operativo gerem o software especializado que gere os recursos do computador de acordo com as necessidade e objetivos do sistema.

T06- construção de sistemas

**Sistemas com um propósito específico:** estes sistemas foram desenhados para fazer funções definidas pelo seu ambiente com pouco ou nenhuma interação com sistemas externos. Opera independentemente com input e outputs controlados e otimizado para segurança eficiência e confiabilidade. Exemplo sistemas de dispositivos médicos, automação industrial etc.

**Sistemas sem SO**: muitos sistemas que estão dentro de dispositivos ioT sistemas de controlo em tempo real indústria, as aplicações são desenhadas para correrem diretamente no hardware. Não usam muitos recursos como CPU e memoria, são mais rápidos a começar a fazer a tarefa e tenho melhores tempo de resposta e tem controlo preciso sobre operações de hardware.

**Desenvolvimento:** código que interage diretamente com o hardware e os endereços de memoria. Utilizar SDKS que dão funções para interagir com o hardware, configurar um compilador e conjunto de ferramentas para fazer com que o código esteja otimizado e compatível com o hardware. Definir o layout de memoria, fazer debug no harware físico pois a integração é feita de forma pequena só com libraries especificas.

**Sistema em tempo real**: desenhado para gerir tarefas que são sensíveis e tem de feitas rapidamente, com restrições de tempo. Este sistema tem benefícios:

* Gere tarefas de modo eficiente: permite multitasking com sobrecarga mínima.
* Confiabilidade: garante alta performance de sistemas críticos.
* Desenvolvimentos em módulos: permite fácil integração com vários componentes e subsistemas.

Este tipo de sistema da:

* Sincronização entre tarefas: as tarefas não interferem umas com as outras
* Comunicação entre tarefas: mecanismos de controlo para troca de dados segura
* Alocação de memoria: assegura que há memoria disponível suficiente enquanto a aplicação está a correr.
* Agendamento determinado: as tarefas são completas no espaço de tempo determinado
* Tratamento de interrupções: interrompe tarefas para fazer tarefas críticas.

Desenvolvimento deste género de sistemas:

O kernal deste sistema juntamente com task and time management e gestão de recursos são compilados e conectados todos juntos dentro da aplicação num só executável que é carregado do hardware. O caso deste sistema são os comandos de um avião.

Sistemas para propósitos gerais: sistemas versáteis para poderem fazer diferentes tarefas utilizando muitas aplicações. Capaz de correr vários tipos de software, desenhados para interagir com vamos periféricos, sendo que podem ser usados para word processing, web browsing, data analysis, gaming, and more.

Benefícios: flexíveis, capazes de interagir com o utilizador, funcionam em diferentes campos de utilização.

Desafios:

* Imprevisíveis: não dá para saber quais dão os programas que vão correr
* Quantidade de programas: a quantidade de programas que vai ser corrido não pode ser predeterminada.
* Utilização de recursos variável: não dá para prever a quantidade de recursos que vão ser usados

Solução:

* Programas correm isolados: os programas só correm na memoria disponível e não podem interferir uns com os outros assegurando estabilidade e execução independente.
* Os recursos do sistema não conseguem ser acedidos diretamente, estes são mediados, atribuídos e controlados pela o sistema

A diagram of a system

Description automatically generated

T07- Sistema operativo

**Objetivos de um sistema operativo**: gerir recursos de forma eficiente, operações do sistema confiáveis, suportar multitasking e simultaneidade de programas, segurança e estabilidade, foco no balancear as necessidades do sistema e utilizador.

Alocação de recursos: gere todos os recursos e revolve conflitos entre os processos para garantir a eficiência e boa utilização de recursos

**Controlo dos programas**: controla a execução dos programas par evitar erros e controla o uso para garantir que o sistema é usado corretamente e seguro.

**Funções orientadas para o utilizador**: interface (GUI e CLI), execução de programas correr e carregar aplicações sem problemas, multitasking e responsividade, armazenamento estruturado e recuperação de informação

* Funções orientadas para o sistema:
* alocação de recursos,
* gerir processos,

proteção e segurança proteger o hardware e o SO com:

* controlo do acesso dos recursos,
* prevenir programas de interagir uns com os outros para não serem corrompidos
* pedir autenticação para dar acesso aos recursos
* bloquear acessos externos

manter contagem: saber o tipo e quantidade de recursos utilizados por cada utilizador, permitir que sejam cobrados valores com base nos recursos usados cloud service e disponibilizar estatísticas para reconfiguração e otimização.

Kernel: componente central que interage com o hardware e que supervisiona operações de low level, inicia as operações dos sistemas ao ser o primeiro programa carregado em memoria tem acesso ao hardware e partes críticas do sistema.

Software reativo e assíncrono do kernel: o SO espera por eventos (ex: input), executa rotinas só quando esses eventos acontecerem, estes eventos ocorrem de forma imprevisível e são feitos imediatamente ou postos na fila de processamento.

**Utilização do kernel:**

Interrupção: sinal do hardware ou software que é preciso atenção imediata (ex: keyboard input, temporizadores, falhas do sistema)

Chamadas do sistema: aplicações que ativam um pedido do SO fazendo com que sejam ativados eventos de rotina, gerando uma interrupção para o kernel fazer um processo.

Modo kernal: aceso total e acesso aos sistemas críticos como a memoria. Este modo permite que sejam corridos todos os processos necessários.

Modo utilizador: acesso limitado a todos os recursos para prevenir que aplicações acedam diretamente ao sistema por exemplo acesso só a parte da memoria alocada. As aplicações correm neste modo

Estes modos são aplicados pelo processador para assegurar a estabilidade e segurança do sistema.

Os modos são mudados um com o outro quando existem interrupções para garantir que a mudança é feita sem problema.

T08- Processos

**O que é um processo**: e um programa que esta a ser executado, um programa é uma entidade estática dentro de um ficheiro que consiste num set de instruções que definem uma sequência de operações para o computador fazer.

Ao contrário dos programas que são estáticos os processos são dinâmicos pois faz operações e interage com os recursos do sistema.

Um Programa torna-se num processo quando é carregado na memoria.

Um processo tem: **parte constante**, as instruções que não mudam, e parte dinâmica informação como variáveis, o programa evolui durante a execução capturando o estado do processo.

Um processo precisa dos recursos do sistema para correr: **memoria** (texto e informação), **tempo do CPU** para executar instruções e computações**, input/output devices** acesso a periféricos, **ficheiros** acesso a informação persistente

O sistema operativo (SO) lida com vários processos usando o a Process Control Block (PCB) para cada processo no sistema, o PCB guarda toda a informação essencial para gerir o processo de maneira eficiente.

As tabelas de processos são a estrutura mantida para o SO saber quais são os programas que estão ativos.

**O PCB tem informação**:

* Identificador do processo (PID): número único que identifica o processo

Único identificador, normalmente com número seriados para garantir eficiência.

Quando o limite de identificadores é atingido sistema volta aos identificadores que estão livres e atribui os novos processos aos valores livres.

* Estado do processo: estado corrente de cada processo

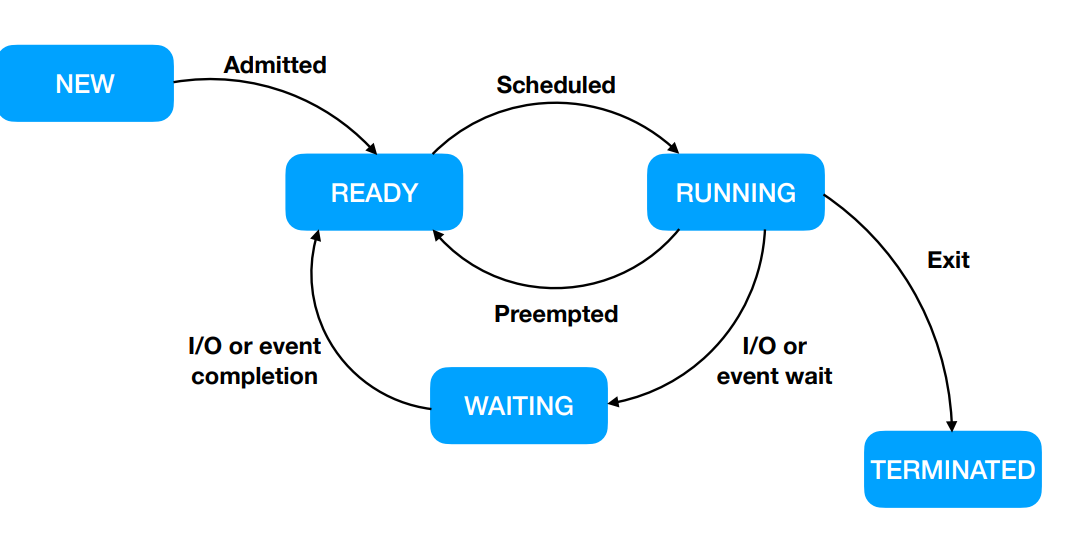
Novo: processo está a ser criado.

A correr: instruções estão a ser executadas pelo processador.

Espera: processo esta a espera de um evento específico

Pronto: o processo está pronto para ser atribuído ao processador

Terminado: acabou a sua execução



Novo 🡪 pronto: o processo é inicializado em que é alocado e carregado a memoria passando a ready/ pronto.

Ready 🡪 running: sleciona o programa e este passa para a correr.

Waitting🡪 ready: quando o evento que o processo esta a espera aconteceu

Running 🡪 waiting: quando o processo esta a espera por exemplo de input do utilizador

Running🡪 ready: para o programa preventivamente para pôr outro programa a correr o programa que foi parado volta para ready.

A correr🡪 terminado: processo está terminado.

* Contador de programas: sabe qual é o próximo programa.
* Registos do CPU: registo de todos os processos específicos
* CPU Scheduling: sabe as prioridades das tarefas
* Gerir memoria: detalhes da alocação ao processo
* I/O status: dispositivos de input output alocados e lista de ficheiros abertos
* Informação dos recursos: utilização do CPU, tempo passado no processo e tempo limite.

**Process scheduling overview:**

Para maximizar a utilização do CPU e a mudança rápida de processos este agendamento de processo decide que programa vai ser executado depois no CPU.

Existem **3 filas principais**:

* Fila dos processos: inclui todos os processos do sistema
* Fila de pronto: processos prontos e a espera de serem executados
* Fila de dispositivos: processos que estão a espera para aceder a um dispositivo específico.

**Mudança de contexto**: aconteceu quando o CPU muda para um processo diferente, o sistema deve gravar o estado do processo que foi substituído e carregar o estado do processo do processo novo.

O estado dos processos esta guardado no PCB, em que os valores que são guardados são o contador, stack pointer e registos específicos do CPU

A diagram of a computer program

Description automatically generated

Esta mudança de contexto tem impactos: faz com que o sistema não seja produtivo durante a mudança, a complexidade do SO e o tamanho do PCB influenciam a duração da mudança.

Para maximizar a utilização do CPU o sistema deve encontrar um balanco estre as trocas de contexto:

Se for **muito frequente** o sistema perde muito a gravar e carregar programas e pouco tempo a fazer os processos

Se for **pouco frequente**: pode dar pouco trabalho ao CPU especialmente em time sharing em que vários processadores competem pela utilização do CPU.

Um equilíbrio garante que o CPU é utilizado de forma eficiente mantendo a responsividade e a escolha correta de processos.

Tipos de Agendamento

* CPU Scheduler (pequena duração): seleciona, executa e aloca o processo. Único processo de agendamento de processo em sistemas mais simples, invocado frequentemente
* Job Scheduler (longa duração): escolher o processo da fila dos processos pronto a ser executados, não é invocado frequentemente pode ser feito mais lentamente.
* Duração media: conhecido como swapping (remover o processo do disco para reduzir o carregamento da memoria e melhorar a eficiência do CPU).

Swap-in: restaura os processos suspendidos para a fila dos processos prontos, do disco para a memoria, quando os recursos do sistema são insuficientes.

Swap-out: move processo de waitting ou ready para suspensos (memoria para disco) para libertar recursos do sistema.

Disk thrashing: acontece quando o sistema passa mais tempo a mudar programas entre o disco e a memoria do que a fazer processos levando a declínio da eficiência do sistema

T09-Memory management. CPU scheduling.

Alocação de memoria:

* Alocação dinâmica: adaptação a processos com variação de necessidades.

Os **desafios** desta alocação são: fragmentação, com os processos terminam a memoria disponível fica fragmentada em blocos livres. Complexidade, gerir a memoria livre para novas alocações é desafiante.

A solução: **paging** a memoria é dividida em blocos fixos as páginas (memoria logica) e **frames** (memoria física)

Memoria logica: o computado vê a memoria logica como um grande processo com endereços lógicos seguidos.

Memoria física: são sítios de memoria dentro do computador que podem estar separadas em frames.

* Isolamento: garantir que os processos não podem aceder e modificar a memoria uns dos outros ou aceder ao kernel Memory.

Proteção de memoria e isolação de processos:

A isolação de processo como já foi dito faz com que cada processo trabalhe no seu espaço virtual, prevenindo que processos interfiram uns com os outros. um sistema definido por privilégios garante que os processos não possam aceder a memoria do kernel. Se os processos acederem sem autorização pode criar exceções como segmentação e erros.

Fatures de proteção de memoria: tal como em Linux, em que precisamos de chmod u + x para dar permissão os scripts para executar aqui funciona da mesma forma, existem bits que dão permissões ao as restringe. O SO (sistema operativo) aplica estes bits de proteção durante a tradução do endereço

* Mapeamento: como mudar entre os programas compilados (endereço logico) e a memoria física.

Memoria virtual: estende a quantidade de memoria para alem do espaço físico para alocar mais processos, cria um espaço virtual que cada processo tem o seu e espaço virtual.

**Tradução dos endereços**: o MMU ou e Memory Management Unit pertence ao CPU e traduz os endereços virtuais para endereços físicos usando tabelas de páginas guardadas na RAM. Cada processo tem a sua tabela de página gerida pelo kernel como parte do PCB Process Control Block.

**O Processo de tradução**: durante as mudanças o kernel da update no MMU usando as tabelas de páginas para ativar o processo. Para acelerar as traduções o MMU usa a Translation Lookaside Buffer (TLB) para ver os mapeamentos mais recentes.

**Partes privadas da memoria**: cada processo mantém a sua própria informação como variáveis, isto garante que o contexto do processo é único e definido pela sua própria informação apesar de poder partilhar código com outros processos

**Alocação de memoria partilhada**: processos a correrem o mesmo programa partilham o executável, reduz o uso da memoria e aumenta a eficiência.

**Scheduling:** este processo define como o CPU e outros recursos são alocados a processos ao longo do tempo. Os objetivos do scheduling são:

1. Maximizar a utilização do CPU
2. Garantir o uso igual entre os processos
3. Ser responsivo para os utilizadores

Critérios de scheduling:

* Utilização do CPU: maximizar a sua utilização durante a maior quantidade de tempo possível.
* Throughput: maximizar o número de processos completos por unidade de tempo
* Turnaround Time: minimizar o total de tempo que leva a executar o processo de início ao fim.
* Tempo de espera: minimizar o total de tempo que os processos estão em espera
* Tempo de resposta minimizar o tempo de resposta

O scheduling deve estar alinhado com os propósitos gerais de um sistema priorizando reponsividade e quantidade de programas feitos (Throughput). Sistema em tempo real, priorizam a previsão e deadlines.

CPU scheduler (Dispacher): sleciona processos da file dos processos que estão prontos, estes são alocados ao CPU. Então o scheduling ocorre o processo passa por estas fases:

Running/ a correr para waiting por uma chamada do sistema

A correr para pronto /ready por preemption

Waiting para pronto /ready por interrupção

E terminado por chamada do sistema.

Cooperative scheduling: o processo mantém o CPU a acabar de completar a sua tarefa

Preemptive scheduling: o sistema para parar um processo para fazer outro aumentado a responsividade.

Scheduling serve para: multitasking, boa distribuição da utilização de recursos, resposta rápida ao input do utilizador.

**Round robin:** processos são agendados numa fila circular, em que cada processo tem uma fatia do círculo, são simples, rápidos e responsivos.

No Linux: o round robin quantificação das tarefas pela prioridade e quantidade, os processos são corridos pela quantidade menor de tempo que demoram quanto menos mais a probabilidade tem de começar por esse processo.

Serve para **propósitos gerais** com trabalhos mistos, tem **boa escalabilidade** para um grande número de tarefas por causa dos seus ajustes dinâmicos. Minimiza que processos acabem preventivamente.

**Agendamento em tempo real**: este tipo de agendamento requer garantias nas entregas dos deadlines dos processos e que tenham comportamento previsível.

Rate Monotonic Scheduling (RMS): propriedades estáticas se o processo demorar menos tem maior prioridade,

Earliest Deadline First (EDF): propriedades dinâmicas, deadline mais próximo maior a prioridade, se houver overload as tarefas ariscar perder o seu deadline.

Multilevel scheduling: agendamento que agrega o processo em diferentes filas baseado em características como a prioridade, tipo de processo e necessidade de recursos.

Tipos de multilevel:

Fixo: processo são atribuídos a uma fila com base em critérios predefinidos

Dinâmico: processo podem se mover entre filas com base no seu comportamento.

Vantagens do multilevel:

* Agendamento a medida para diferentes tipos de trabalhos
* Melhor responsividade para processos interativos.

Desvantagens do multilevel:

* Pode levar a que as tarefas com pouca prioridade não sejam feitas se não for gerido de maneira apropriada
* Grande complexidade comparada com uma fila de agendamento de tarefas

Agendamento em muito processadores

Agendamento global: os processos estão numa fila única e qualquer processador pode fazer a tarefa.

* Vantagens: mais bem balanceado entre processadores
* Desvantagens: grande disputa na fila partilha dos processos

Agendamento dividido: filas separadas cada uma atribuída a um processador

* Vantagens: implementação simples e evita migrações
* Desvantagens: potencial para trabalho desbalanceado se as tarefas não estiverem bem distribuídas.

Agendamento misto: combina os anteriores e os processos podem migrar entre diferentes processadores.

* Vantagens: faz com que as tarefas estejam bem distribuídas e reduz a migração de processos
* Desvantagens: complexo para implementar

Tipos de escalonamento:

FirstCome FirstServed: Primeiro a chegar vai ter o primeiro lugar na fila. Tem vantagens: Simples, fácil de implementar, não é preemptivo Running para ready. Tem problemas, baixa utilização do CPU

ShortestJobFirst: seleção da fila ready do programa que demora menos tempo, Não é preemptivo, pois se chegar um processo que demore menos tempo não vai ser executado logo, O processo em execução é comutado com o processo novo se o tempo restante de execução for maior que o tempo de execução do processo novo (Shortest-Remaining-Time-First (SRTF))

Roundrobin: Adptado para sistemas partilhados multiutilizador cada processo obetem uma unidade de tempo em forma de fatia de um bolo sendo depois adicionado a fila ready, Se o processo ulttrapssar o seu tempo atribuído o So seleciona o processo seguinte

Processo de swapping:

1. **Identificar processo que tem de ser movido:** se a memoria estiver cheia. O SO escolher um programa inativo ou de baixa prioridade para ser transferido para a memoria secundaria.
2. Processo movido para a memoria secundaria (o disco)
3. Carregar novo processo
4. O processo no disco pode voltar a ser movido para a RAM

**Vantagens:**

* Permite que o sistema operacional execute mais processos do que a memória física disponível suporta.
* Aumenta a flexibilidade e a capacidade multitarefa.

**Desvantagens:**

* A leitura e escrita na memória secundária são muito mais lentas do que na RAM, o que pode causar uma diminuição significativa no desempenho se o swapping for muito frequente (conhecido como **thrashing**).

Sistema MultiProgramado: permite a execução simultânea de vários processos, num único PC.

O rpocesso dos sistemas multiprogramados é:

1. O carregamento de vários processos ao mesmo tempo na memoria principal em que cada processo ocupa uma parte da memoria
2. Scheduling/ Escalonamento de processos o CPU seleciona um processo para ser executado. Quando esse processo entra em estado de espera (por exemplo, aguardando entrada/saída), a CPU passa a executar outro processo pronto para execução.
3. Permite **troca de contexto** em que quando um programa esta a ser executado e tem de ser interrompido o sistema guarda o seu estado atual e carrega o processo que precisa de ser executado

Vantagens:

* Partilha de forma eficiente os recursos
* Otimização do uso do CPU
* Maior produtividade

O PCB contem informações como:

* **Informações sobre o estado do processo** (executando, pronto, bloqueado, etc.).
* **Registros de CPU** (como o contador de programa, registradores e informações do processo).
* **Gerenciamento de memória**, incluindo a lista de blocos de memória que o processo ocupa.
* **Lista de arquivos abertos**.
* **Informações sobre as permissões e privilégios do processo**.
* **Tempo de CPU consumido**.
* **Informações sobre sinais e controle de processos filhos (se houver)**.

Multitasking: sistema operativo executa vários processos de maneira **aparente simultânea**, intercalando a execução entre os processos (mesmo em sistemas com um único processador)

O ciclo **fetch-decode-execute** é o processo básico de operação de uma CPU:

* **Fetch (buscar):** A unidade de controlo lê a próxima instrução da memória (no endereço indicado pelo program counter).
* **Decode (decodificar):** A unidade de controlo analisa o opcode da instrução para determinar que operação deve ser realizada.
* **Execute (executar):** A CPU executa a operação especificada pela instrução.

No **contexto switch** o SO guarda o estado do processo e o estado do processador

**Multiprogramação**: Possível em sistemas com uma única CPU.

**Paralelismo**: Necessita de múltiplas CPUs ou múltiplos núcleos (não ocorre em sistemas com uma única CPU).

* O **escalonador de curto prazo** decide qual processo será executado pela CPU em cada ciclo.
* O **escalonador de longo prazo** gerencia quais processos entram ou saem do conjunto de processos ativos.
* O escalonador de **curto prazo** é executado frequentemente, enquanto o de **longo prazo** é executado com menos frequência.