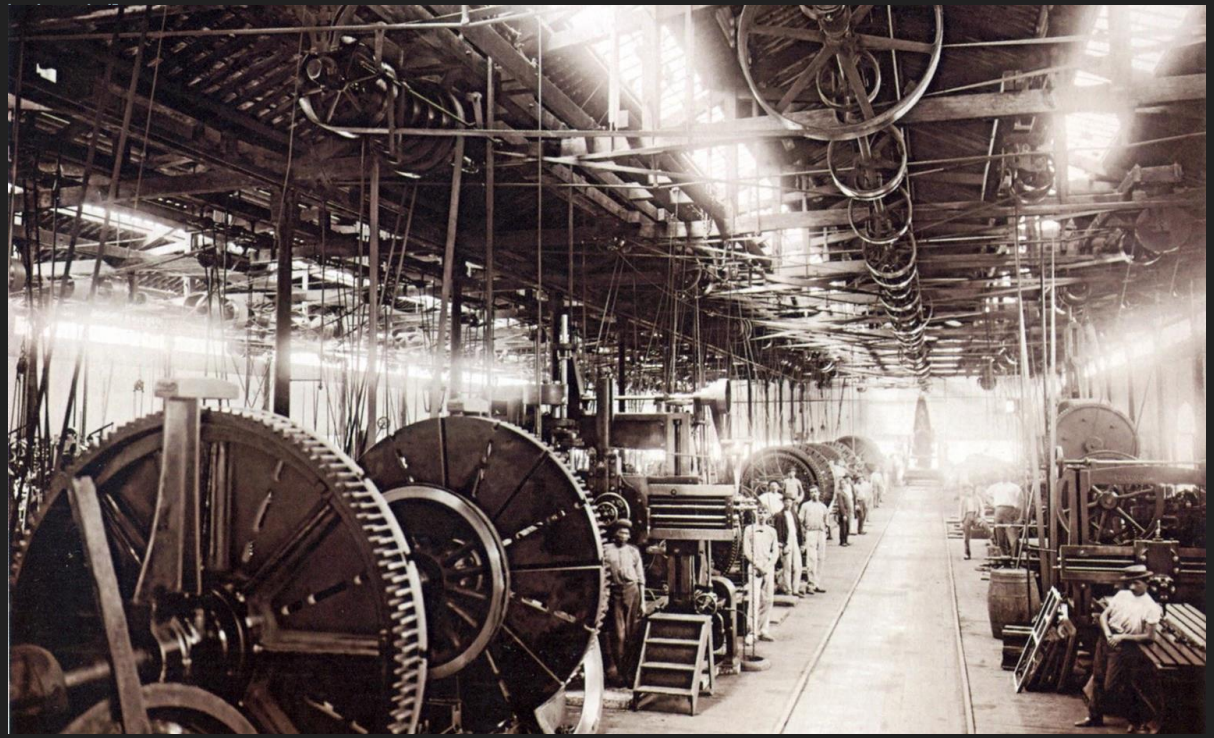
Seminário SO
Escalonamento de Processos

FILIPE CORREIA BELFORT, JOSIEL COSTA DOS SANTOS JUNIOR E STEVEN ROGER DOS SANTOS SOARES

Revolução industrial

[Figura: Representação de um ambiente de trabalho no período da revolução]



Fonte: EDUCA+BRASIL, 2018

XVIII - XIX

Período de grandes inovações

Aumento populacional

Transformações de ordem social e econômica

Máquina de escrever

[Figura: Representação de uma máquina de escrever]

Fonte: GUIA 55, 2023

1575 - Rampazetto

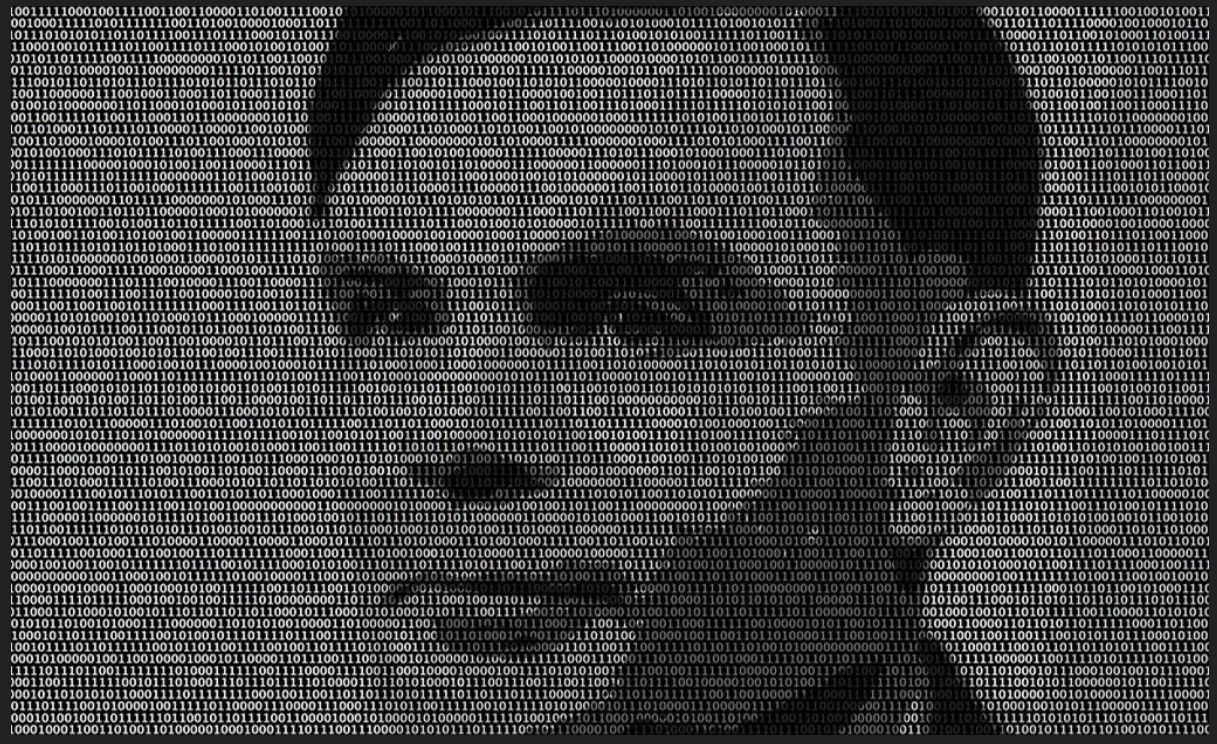
1829 - William Austin

Primeiro mecanismo de escrita cuja invenção foi documentada

No século XX transformou-se em um objeto comum nos escritórios

Alan turing

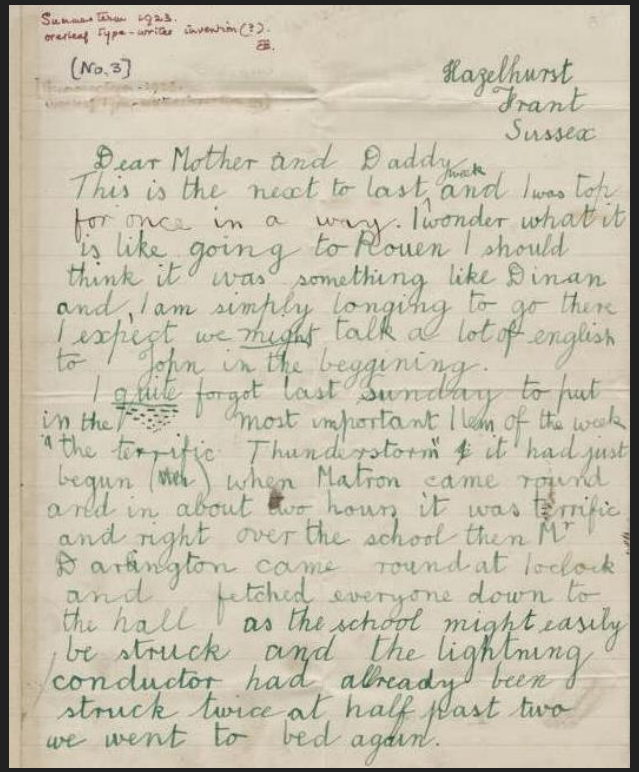
[Figura: Ilustração de alan turing]



Fonte: BBVA Openmind, 2015

Carta de turing

[Figura: Representação da carta de turing aos pais]

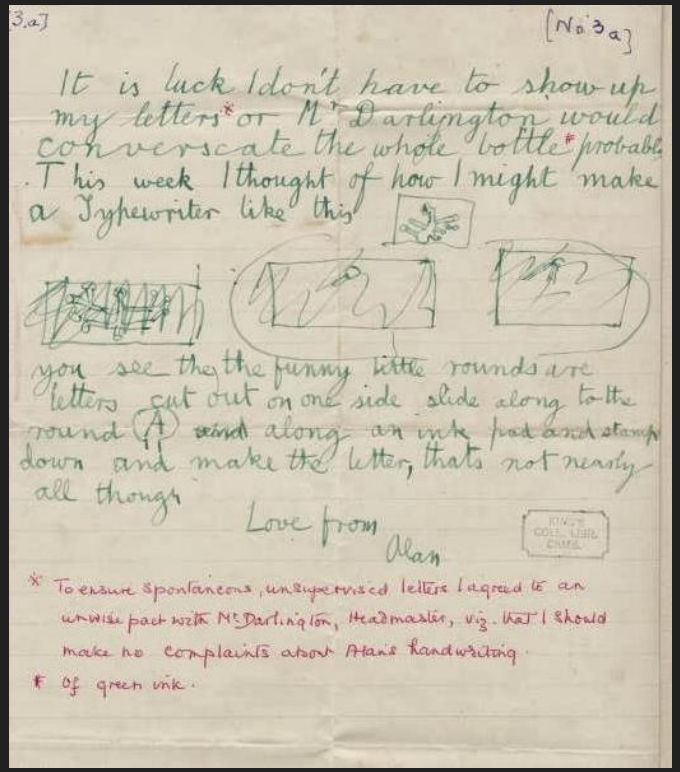


Summer Term 1923.
overleaf Type-writer invention(?).
EB.

(No. 3)

Hazelhurst
Frant
Sussex

Dear Mother and Daddy,
This is the next to last week and I was top for once in a way. I wonder what it is like going to Rouen I should think it was something like Dinan and I am simply longing to go there I expect we might talk a lot of english to John in the beggining.
I quite forgot last Sunday to put in the most important item of the week the terrific Thunderstorm & it had just begun when Matron came round and in about two hours it was terrific and right over the school then Mr Darlington came round at 1o'clock and fetched everyone down to the hall as the school might easily be struck and the lightning conductor had already been struck twice at half past two we went to bed again.



3.a]

[No. 3a]

It is luck I don't have to show up my letters* or Mr Darlington would conversate the whole bottle* probable. This week I thought of how I might make a Typewriter like this

you see the the funny little rounds are letters cut out on one side slide along to the round (A) and along an ink pad and stamp down and make the letter, thats not nearly all though

Love from
Alan

KING'S COLLEGE CAMB.

* To ensure spontaneous, unsupervised letters I agreed to an unwise pact with Mr Darlington, Headmaster, viz. that I should make no complaints about Alan's handwriting.
* Of green ink.

Fonte: HODGES, Andrew. Alan Turing: The Enigma. 1983.

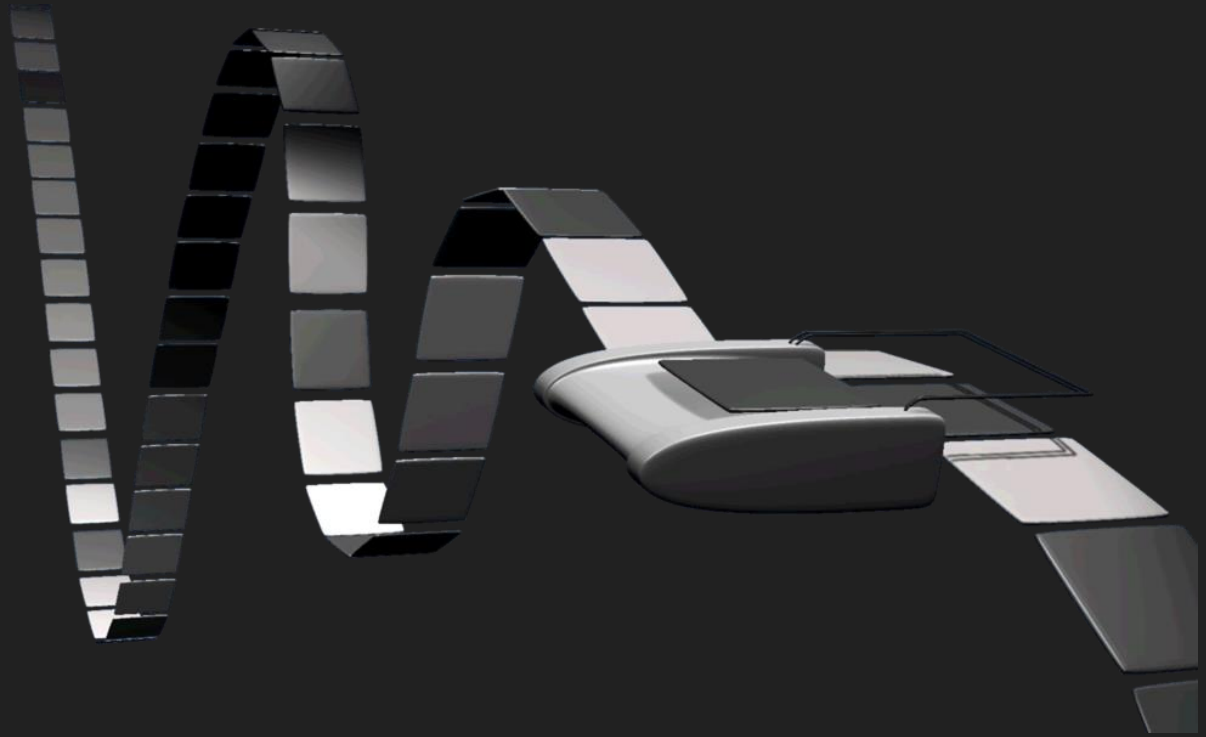
Fascínio pela máquina de escrever de sua mãe

“Projeta” sua primeira máquina

Primeiras noções mecânicas

Máquina de turing

[Figura: Ilustração da máquina de turing]



Fonte: La Naturaleza del Software, 2017

Máquina teórica

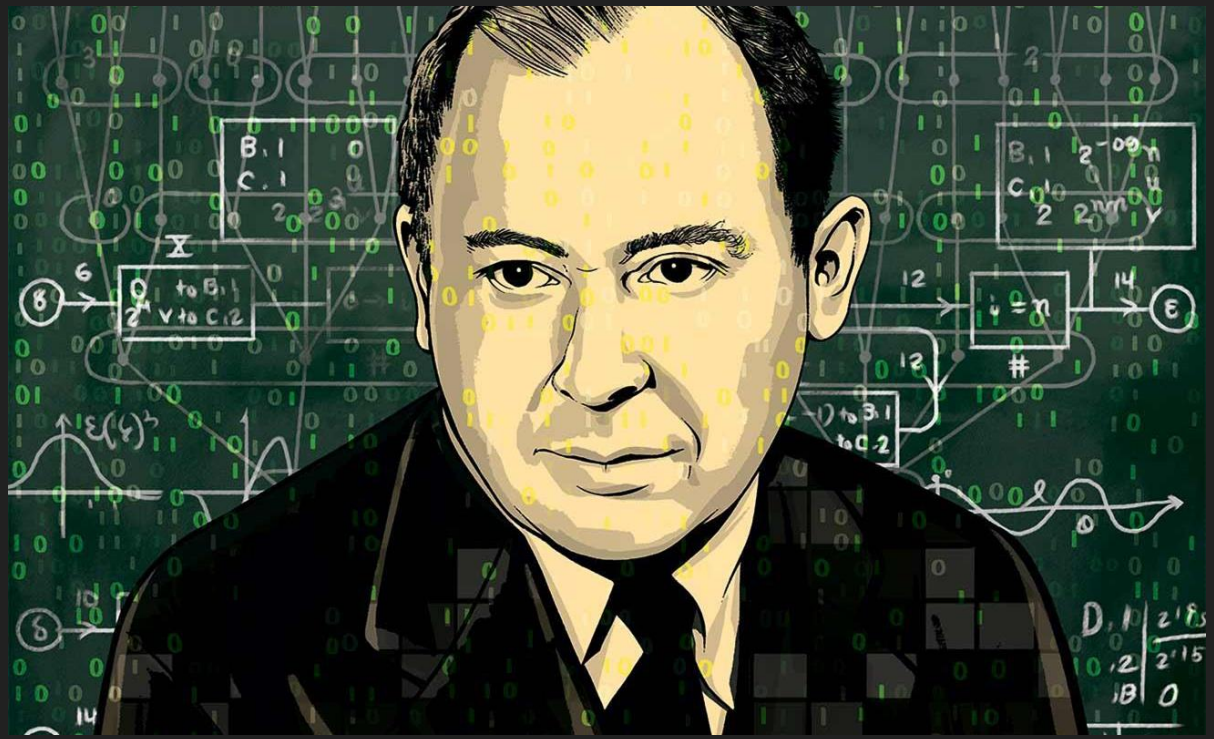
Turing Complete

Funções computáveis

Dispositivo computacional com capacidade de computação máxima

John Von Neumann

[Figura: Ilustração de John Von Neumann]



Fonte: THE NATION, 2022

Aos 18 anos ele já era reconhecido como matemático

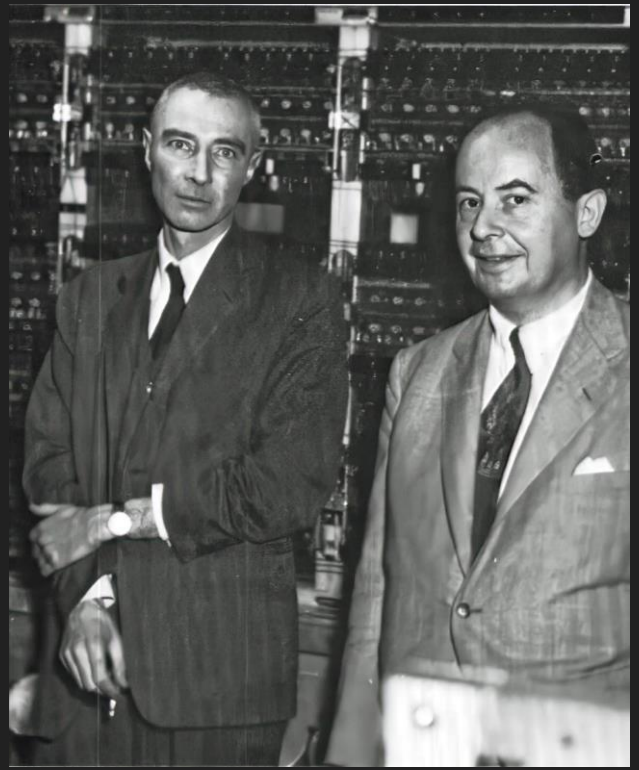
Teoria dos conjuntos e lógica

estabeleceu “framework” matemático para mecânica quântica

Imortalidade para a matemática

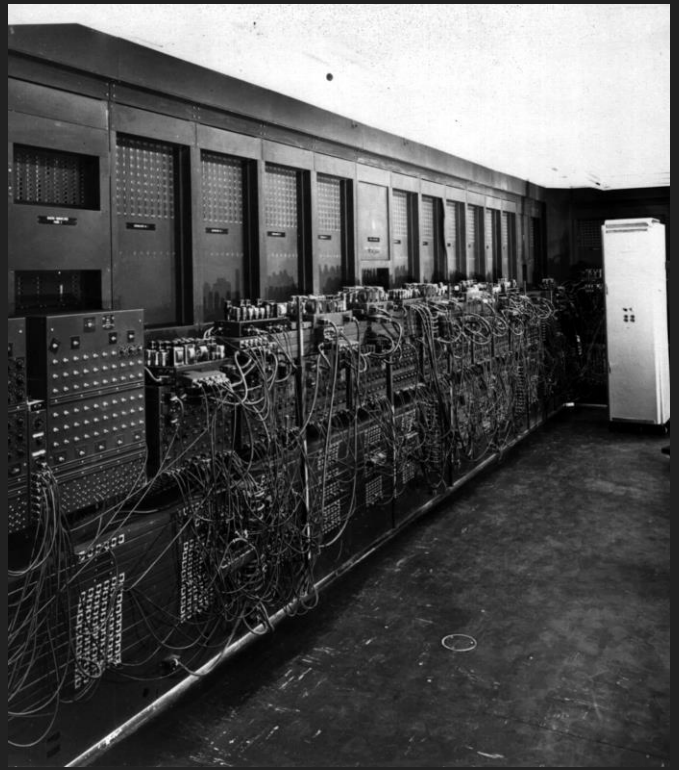
John Von Neumann

[Figura: Von Neumann e oppenheimer no projeto Manhattan]



Fonte: Atomic Heritage Foundation, 2014

[Figura: Fotografia do Computador ENIAC]



Fonte: sims lifecycle services, 2022

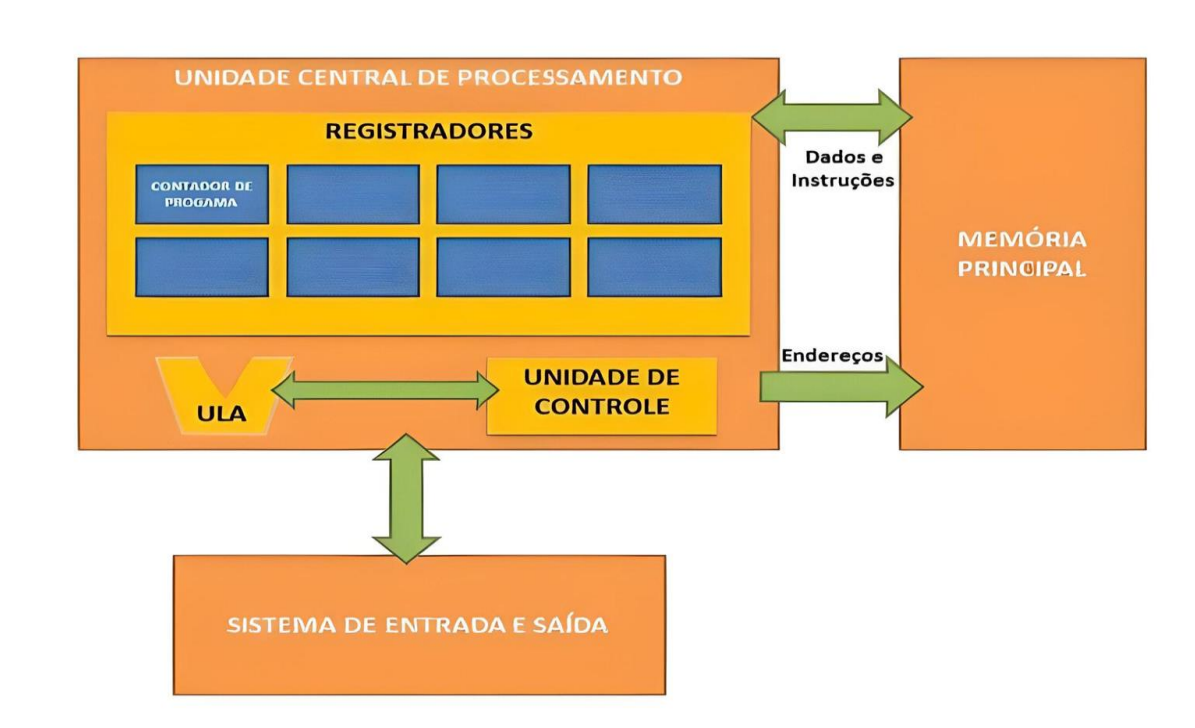
Contribuição no projeto manhattan

Colaboração no ENIAC

Problematização da arquitetura

A arquitetura de Von Neumann

[Figura: Representação da arquitetura de Von Neumann]



Fonte: UNIVERSIDADE SAGRADO CORAÇÃO, 2012

Gargalo de Von Neumann

CPU é responsável por processar os dados

Memória é a responsável por armazenar os dados

O gargalo ocorre na diferença de velocidade que ocorre entre a capacidade de processamento da CPU e o fornecimento dos dados por parte da memória

Ociosidade do processador

Escalonamento

por Steven Roger S.S.

O que são processos?

Breve Contextualização em Código Java

Programa Armário

[Figura: Codigo Armário, uma analogia a processos]

```java
public class Main {
    public static void main(String[] args) {
        // Instanciando um armario este será o Programa
        ArmarioPotes armario = new ArmarioPotes();

        // Criando um Pote este será o sub processo
        //Tamanho e Registro serão as informaçoes que um Processo contem
        Pote pote1 = new Pote( tamanho: 10,  registro: 0000001);

        // Guardando os potes no armário
        //Metodo Será uma Thread
        armario.guardarPote(pote1);

```

[Fonte: Imagem, Steven R.]

Muitas das mesmas questões que se aplicam ao escalonamento de processos também se aplicam ao escalonamento de threads, embora algumas sejam diferentes

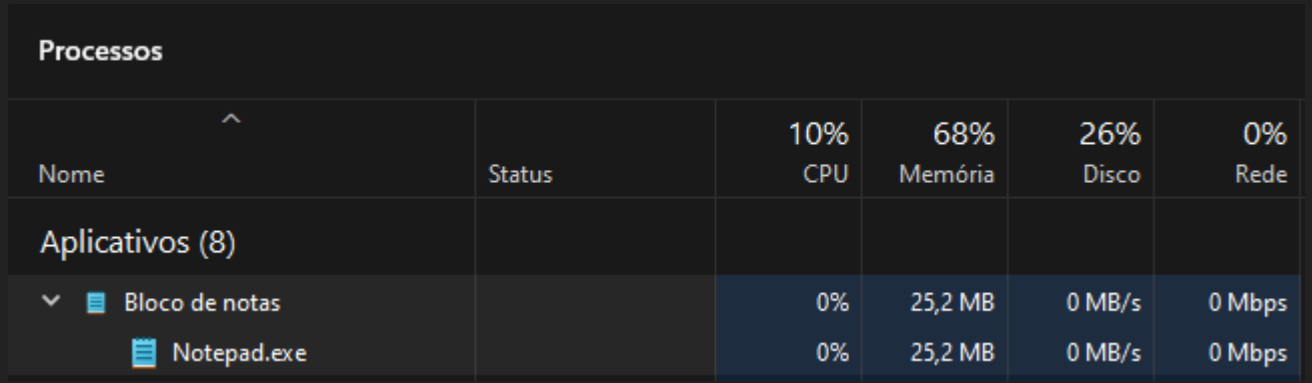
Ou seja…

Processo: Um aplicativo de processamento de texto, essa instância seria o processo global.

Sub-processo: Pode ser visto como uma instância de um objeto dentro desse processo..

Thread: Aqui, a thread seria uma unidade menor de execução dentro desse documento. Pode ser algo como a formatação automática que ocorre enquanto você digita.

[Figura: Gerenciador de Tarefas]



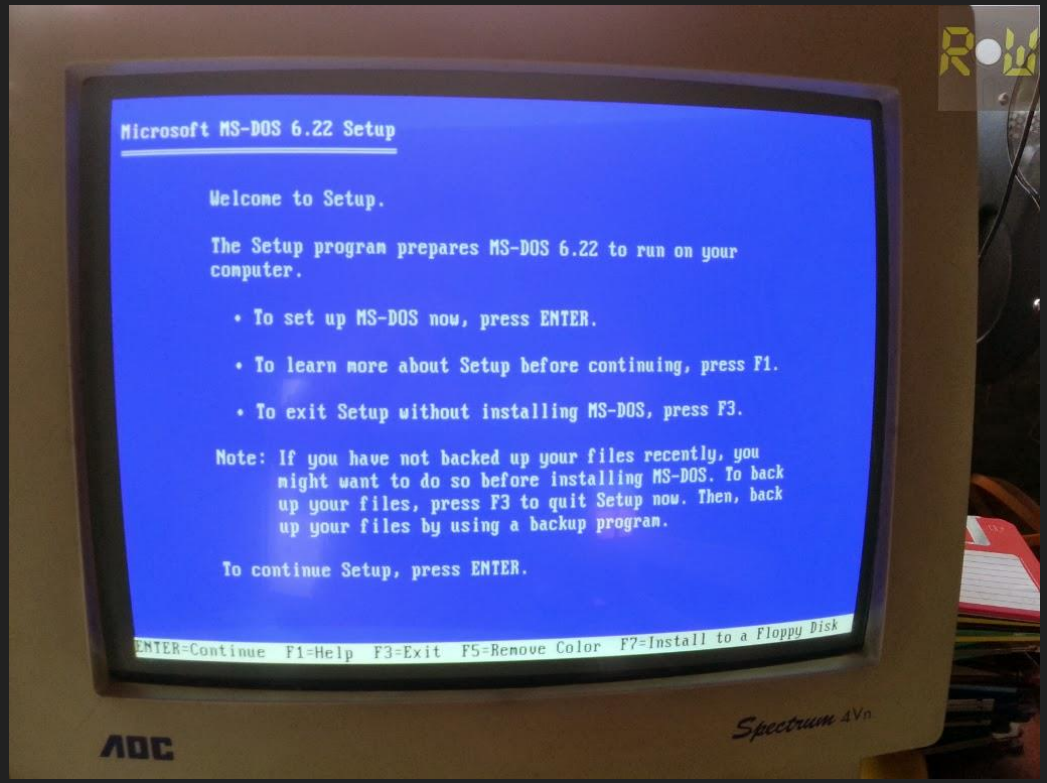
[Fonte: Imagem, Steven R.]

Escalonamento de Processos Para que e Como

Escalonamento de Processos

Agora os PC são multiprogramado, diferente dos monoprogramados como o MC-DOS

[Figura: Tela inicial do programa de instalação do MS-DOS 6.22]



Fonte: Blog do Michael, 2014, Ressuscitando um antigo 286

Escalonamento de Processos

Processos E Threads Concorrentes

Recursos limitados

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**

[Figura: Processos concorrentes (Competição girar a garrafa e para comer Hambúrguer)]

Fonte: Youtube, Canal Super Equipe

Escalonamento de Processos em PC’s atuais

“computadores tornaram-se tão rápidos com o passar dos anos que a CPU dificilmente ainda é um recurso escasso. A maioria dos programas para computadores pessoais é limitada pela taxa na qual o usuário pode apresentar a entrada (digitando ou clicando), não pela taxa na qual a CPU pode processá-la[...]” (Tanenbaum, 2016)

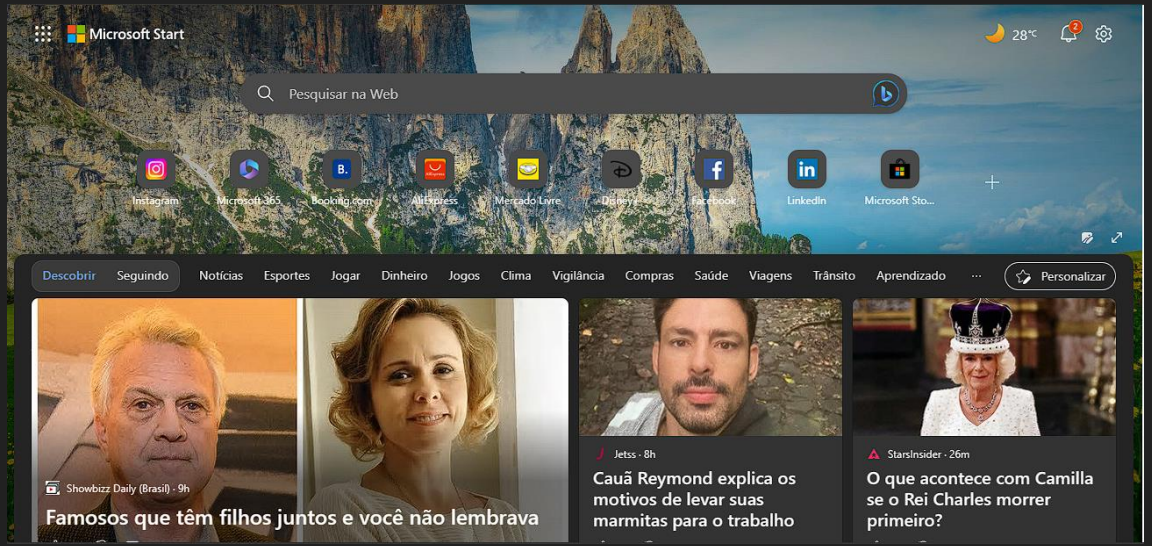
O escalonamento não importa muito em PCs simples

Caso onde o Escalonamento é importante

Servidores em rede

Quando a CPU tem de escolher entre executar um processo que reúne as estatísticas diárias e um que serve a solicitações de usuário via Protocolo HTTP

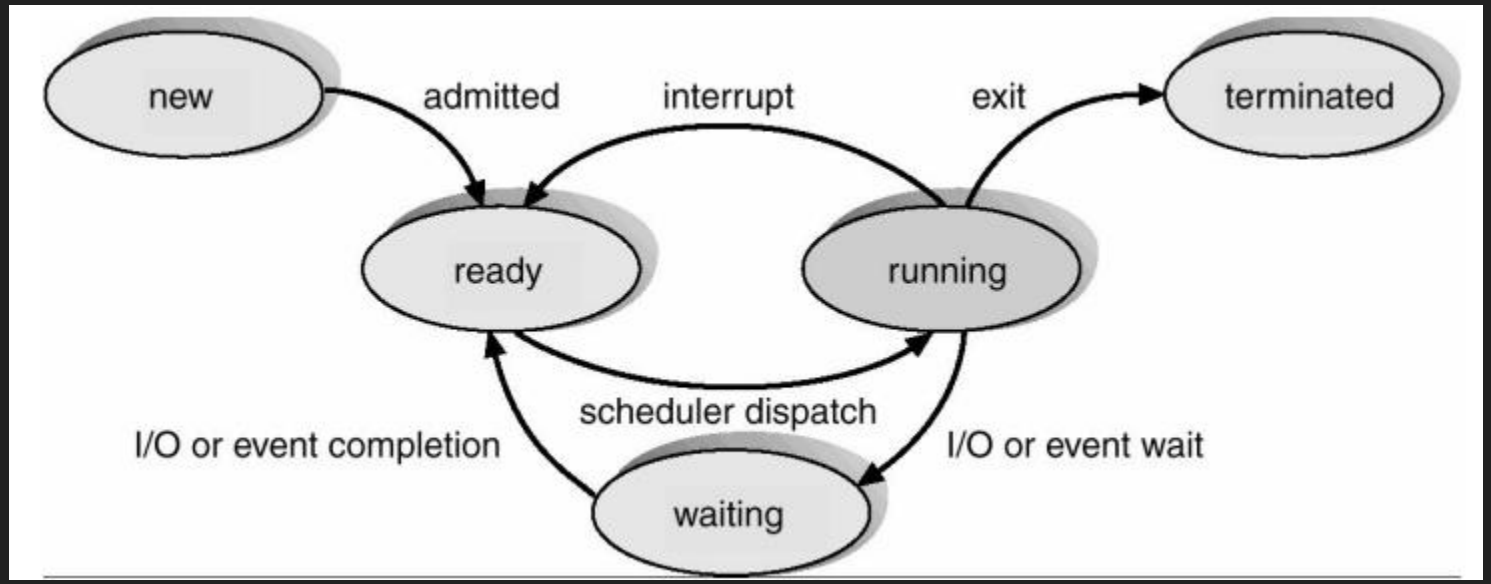
[Figura: Interface do Microsoft Edge]



[Fonte: Microsoft Edge, Steven R.]

Em qual momento ocorre o escalonamento

[Figura: Diagrama de Estados de um Processo]



Fonte: Operating System Concepts and applications Silberschatz, Galvin and Gagne 2002, unit 7

Scheduler : função implementar os critérios da política de escalonamento

Dispatcher: Troca de contexto dos processos após o escalonador determinar qual processo deve fazer uso do processador

Mudança de Contexto: Contexto de hardware e software

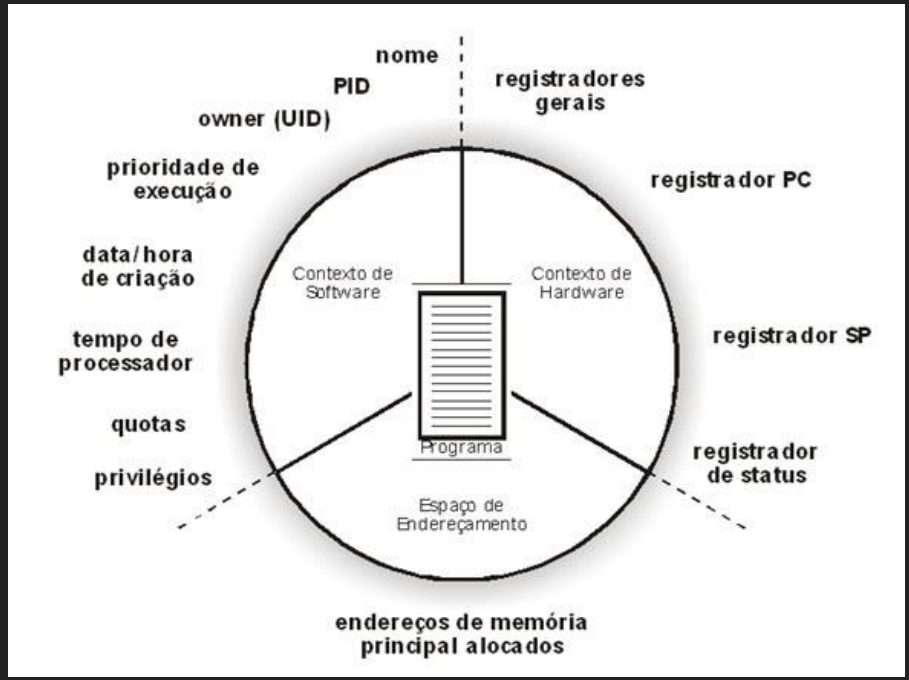
Contexto de Hardware:

- Registradores da CPU (conteúdo dos registradores gerais da CPU e dos registradores específicos)
- Contador de Programa (PC)

Contexto de Software:

- Identificação
- Quotas / Memória (recursos disponíveis para serem alocados em um processo)
- Privilégios

[Figura: Gráfico Contexto do processo]



[Fonte: alexpagernet.blogspot.com,2014,leia um pouco sobre os conceitos básicos de sistemas operacionais]

Mudança de Contexto: Contexto de hardware e software

Além de escolher o processo certo a ser executado, o escalonador também tem de se preocupar em fazer um uso eficiente da CPU, pois o **chaveamento de processos é algo caro**

[Figura: UltraSPARC Microprocesador]

[Fonte: wikichip.org/wiki/sun_microsystems/ultrasparc]

- Os tempos de troca de contexto são altamente dependentes do suporte de hardware

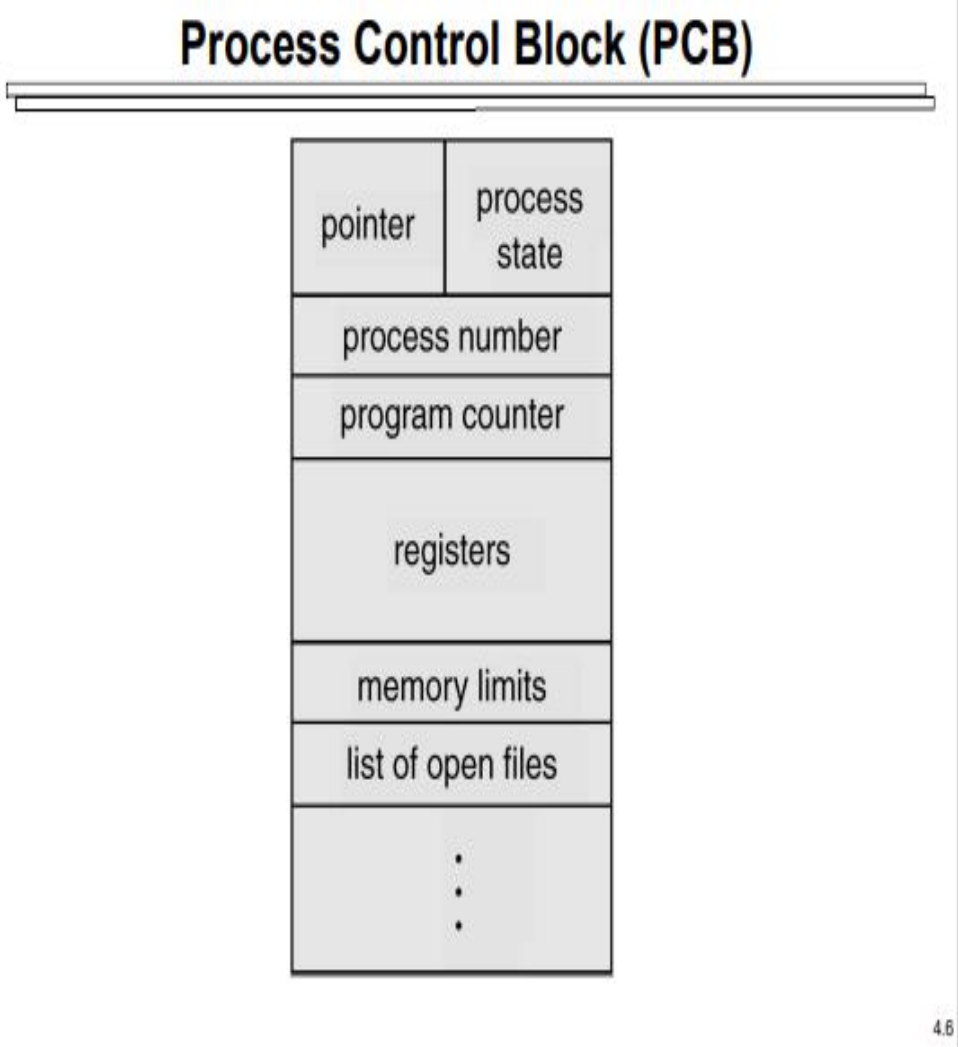
- Ultra Sparc com 144 Registradores

PCB contém informações associadas a cada processo:

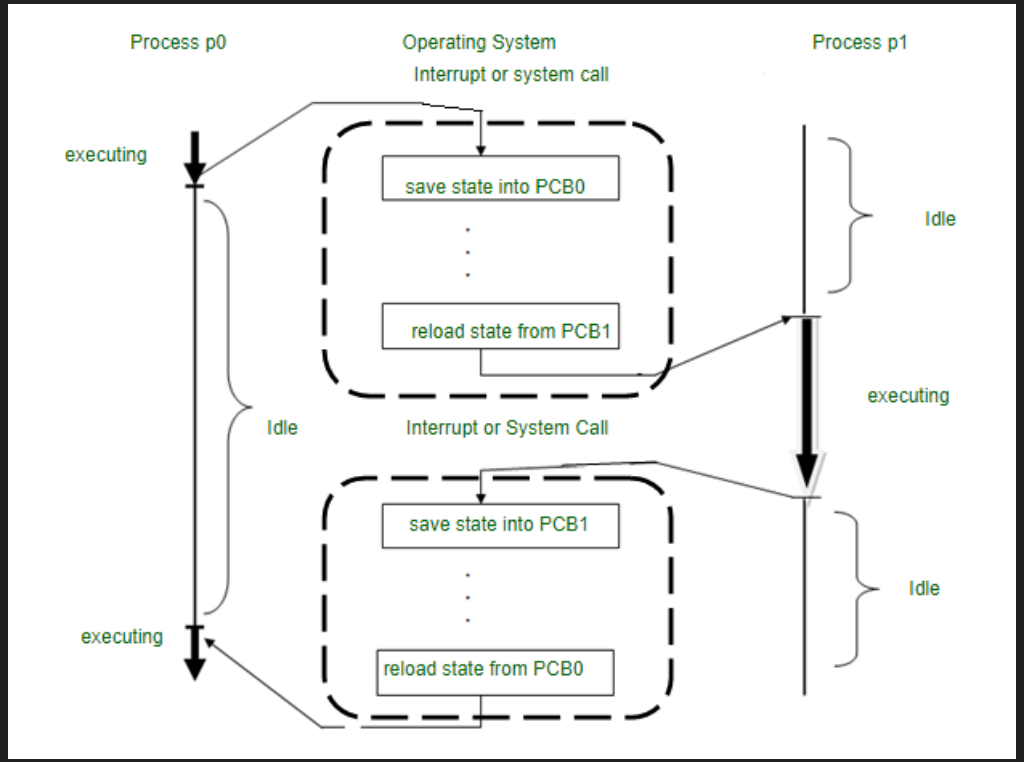
- Estado do processo
- Valor do PC (apontador de instruções)
- Área para guardar valor dos registradores – Infos. para escalonamento de CPU (escalonamento processos)
- Infos. Para gerenciamento de memória
- Infos. De contabilidade dos processos
- Status das operações de I/O (ex.: Infos. sobre arquivos usados)

[Figura: Bloco de controle de processo]

Fonte: Silberschatz, capítulo 2



[Figura: Mudança de Contexto]



[Fonte: geeksforgeeks.org/difference-between-swapping-and-context-switching]

Quem chama o Escalonador?

Interrupção de software (chamada de sistema)

Término
Liberação voluntária da CPU
Operação de E/S
Primitiva de sincronização

Interrupção de hardware

Tempo
Conclusão da operação de E/S

Escalonadores

Não preemptivo ou cooperativo: apenas via interrupção de software
Preemptivo: interrupção de software e interrupção de hardware

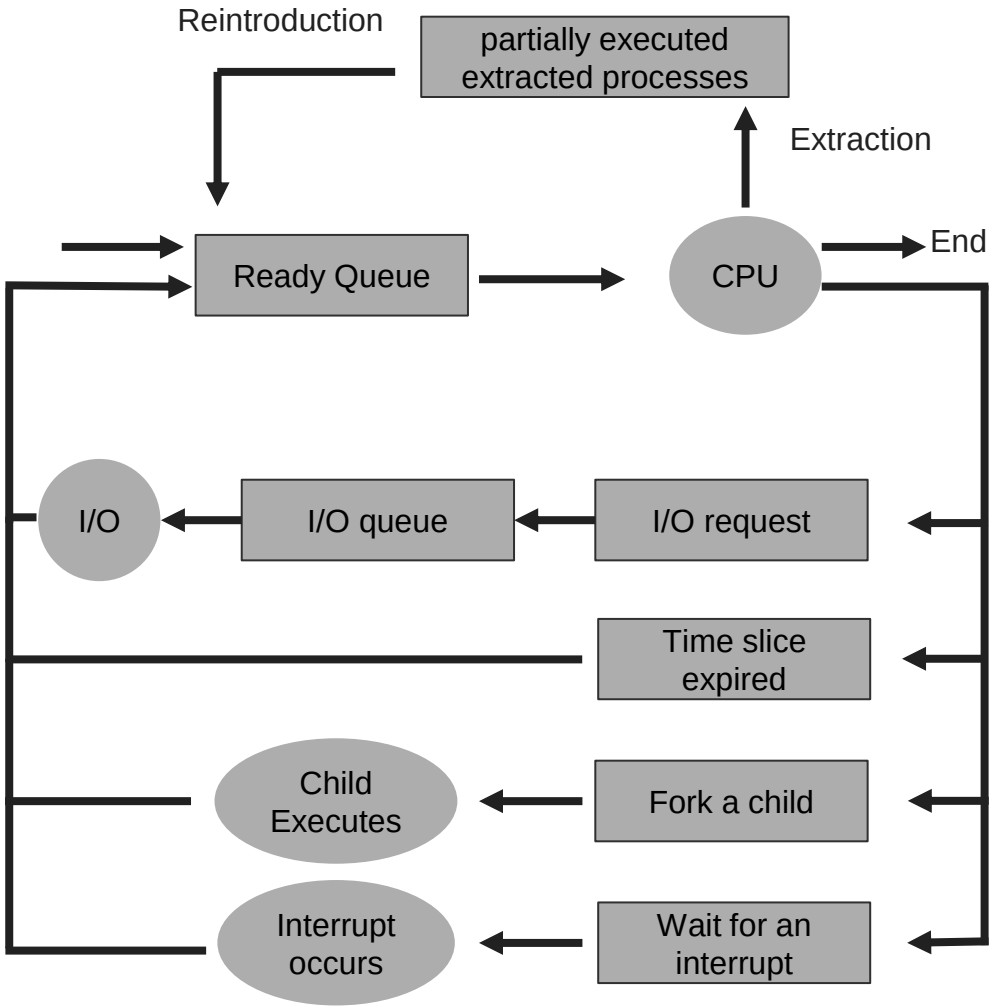
Níveis de Escalonamento

→ De longo prazo (de jobs) - Fila de processos a serem carregados em memória

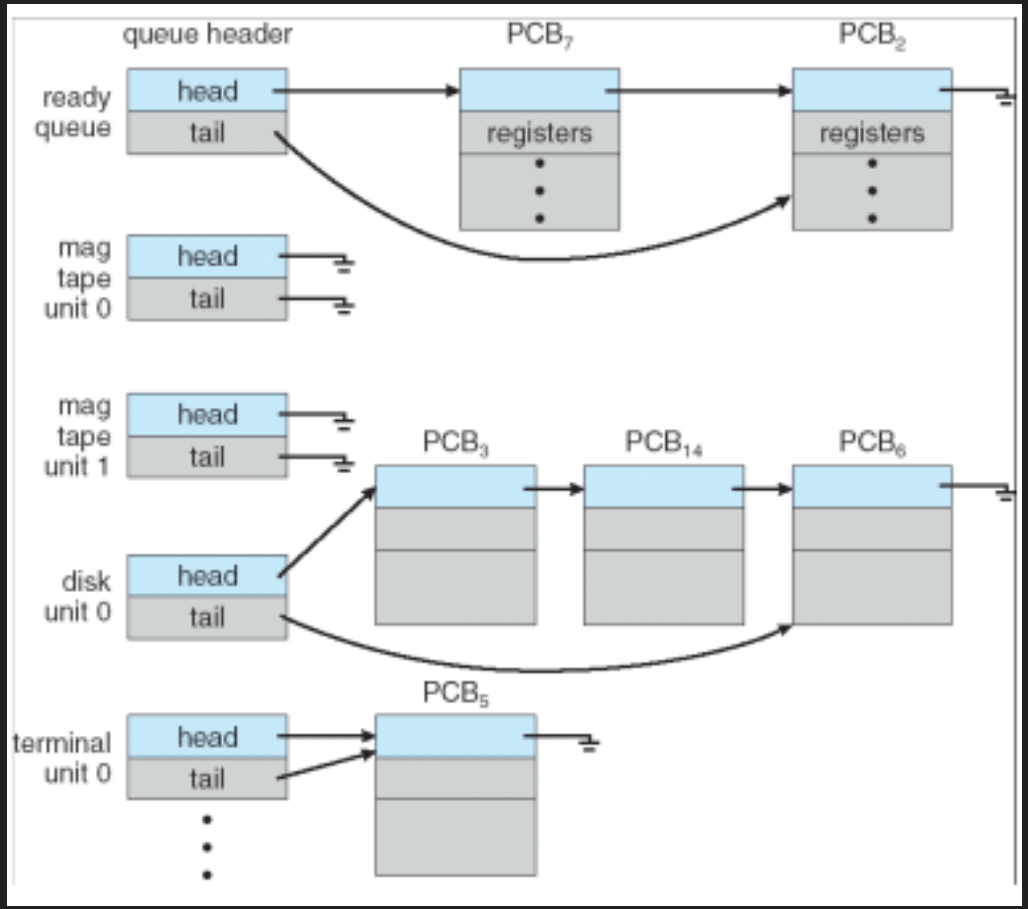
→Em execução - Instruções estão sendo executadas

→ De curto prazo (de CPU)

→ De médio prazo - Sistemas de tempo real - remover processos da memória - reduzir grau de multiprogramação



[Figura: A fila de processos prontos e várias filas de dispositivos de I/O]



Fila de prontos e as diversas filas de dispositivos de I/O

A Tabela de Processos comporta uma fila de nós de **PCB's** encadeados

[Fonte: Operating systems - concepts and applications Silberschatz, Galvin and Gagne, 2001, Cap 4]

Objetivos do algoritmo de escalonamento

dependendo do ambiente

Objetivos

Justiça - dar a cada processo uma porção justa da CPU

Aplicação da política - verificar se a política estabelecida é cumprida

Equilíbrio - manter ocupadas todas as partes do sistema

Todos os Sistemas

Cenários:
Justiça para processos comparáveis

Política local, controle de segurança tem prioridade acima da folha de Pagamento.

Objetivos

Vazão (throughput) - maximizar o número de tarefas por hora. Quantas tarefas por hora saem do sistema

Tempo de retorno - minimizar o tempo entre a submissão e o término. Quanto tempo leva para
receber uma tarefa de volta

Utilização de CPU - manter a CPU ocupada o tempo todo

Sistemas em Lote

Objetivos

Tempo de resposta - responder rapidamente às requisições

Proporcionalidade - satisfazer as expectativas dos usuários

Sistemas Interativos

Cumprimento dos prazos - evitar a perda de dados

Previsibilidade - evitar a degradação da qualidade em sistemas multimídia

Sistemas de tempo real

Tipos de Escalonadores

Preemptivo

- Processo pode ser interrompido antes de terminar ou sem ser bloqueado

- Provoca interrupção forçada para se evitar que um processo tenha 100% da CPU

Não-preemptivo

- Processo nunca é interrompido, execução só para quando processo termina ou é bloqueado

- Um processo pode monopolizar a CPU

Sistemas batch

First In First Out (FIFO)

Shortest Job First (SJF)

Shortest Remaining Time Next (SRTN)

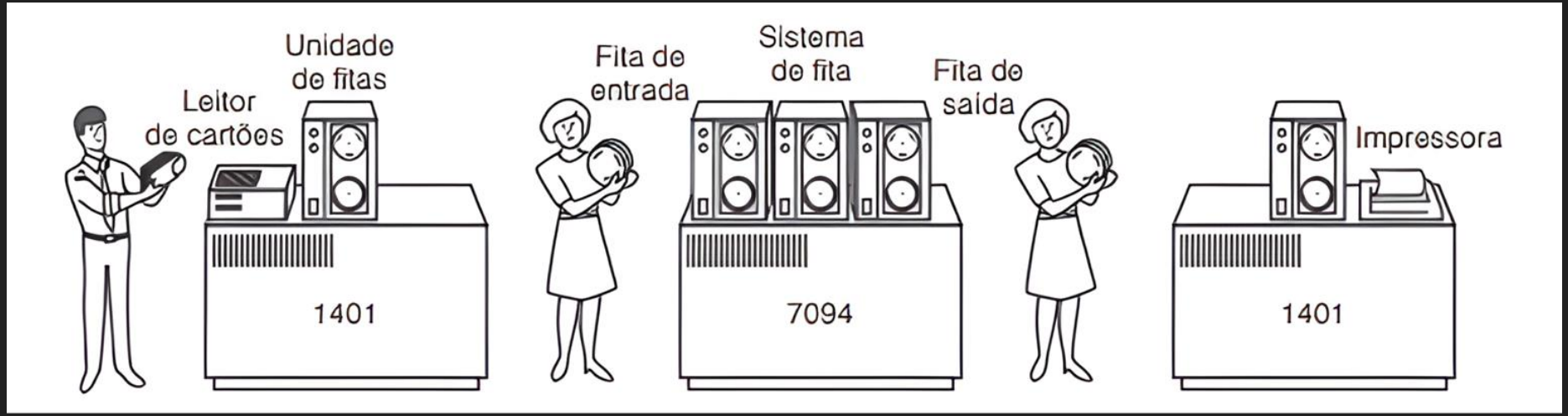
First In First Out (FIFO)

[Figura: Representação de uma fila]

Fonte: Juniata College, 2018

First In First Out (FIFO)

[Figura: Representação de um sistema batch antigo]



Fonte: TANENBAUM, Andrew. SISTEMAS OPERACIONAIS MODERNOS. 4. ed. 2016. 6 p.

First In First Out (FIFO)

Não-preemptivo

O processador é alocado seguindo a ordem de chegada dos processos à fila de processos prontos

O processo que tem a CPU não a libera até que acabe sua execução ou até que fique bloqueado por uma operação de E/S

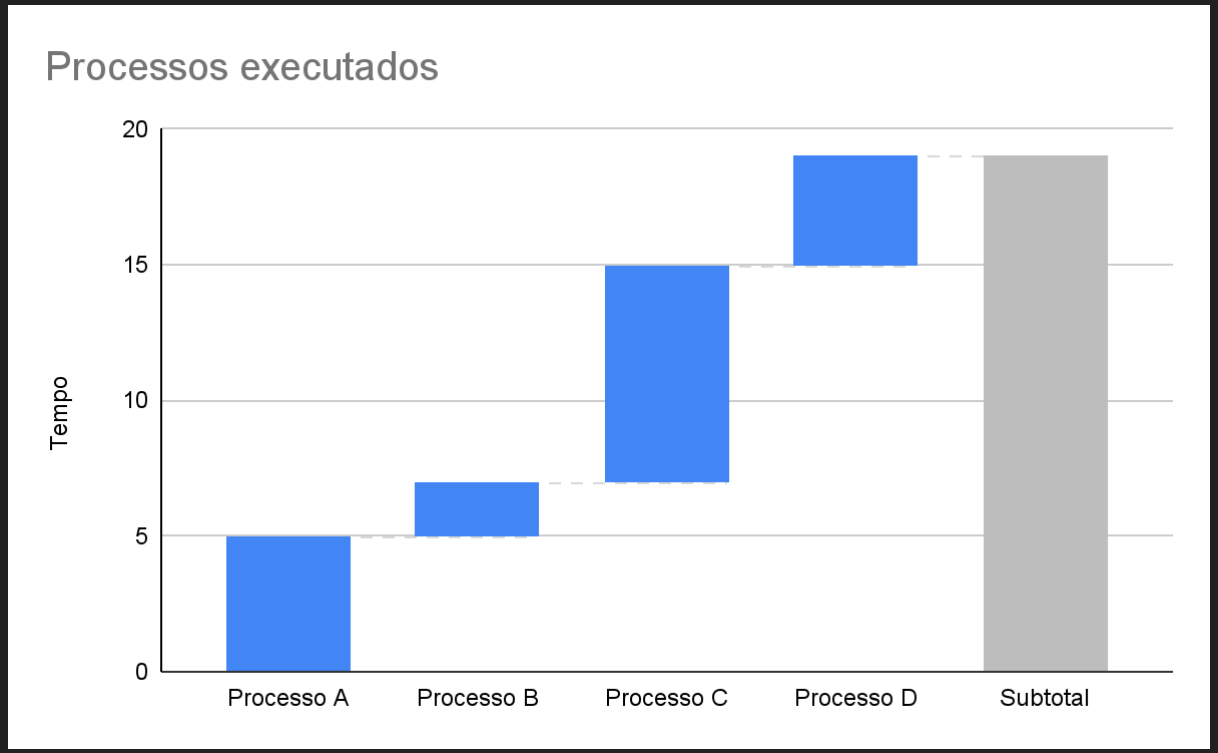
Compreensão intuitiva e de implementação simples

Não é muito eficiente se houver processos curtos esperando atrás de processos longos

First In First Out (FIFO)

Processo	Tempo de processador
A	5
B	2
C	8
D	4

First In First Out (FIFO)



Processo	Tempo de processador
A	5
B	2
C	8
D	4

First In First Out (FIFO)

Processo	Tempo de processador
A	5
B	2
C	8
D	4

Tempo médio de retorno dos processos A, B, C e D:

(5+7+15+19) / 4 = 11.5

First In First Out (FIFO)

```
Função simulateFIFO(fila, tarefas[], numTarefas)
    Para cada tarefa de tarefas[0] até tarefas[numTarefas-1] faça
        Enfileirar(fila, tarefa)

    Enquanto o tamanho da fila for maior que zero faça
        Se tarefa não for nula então
            ExecutarTarefa(tarefa)
            Desenfileirar(fila)
        Fim Se
    Fim Enquanto
Fim Função
```

Shortest Job First (SJF)

Não-preemptivo

Considera-se o tempo de execução de cada processo para alocar ao processador

Pode ser prejudicial se houver muitos processos curtos, pois processos longos podem esperar indefinidamente

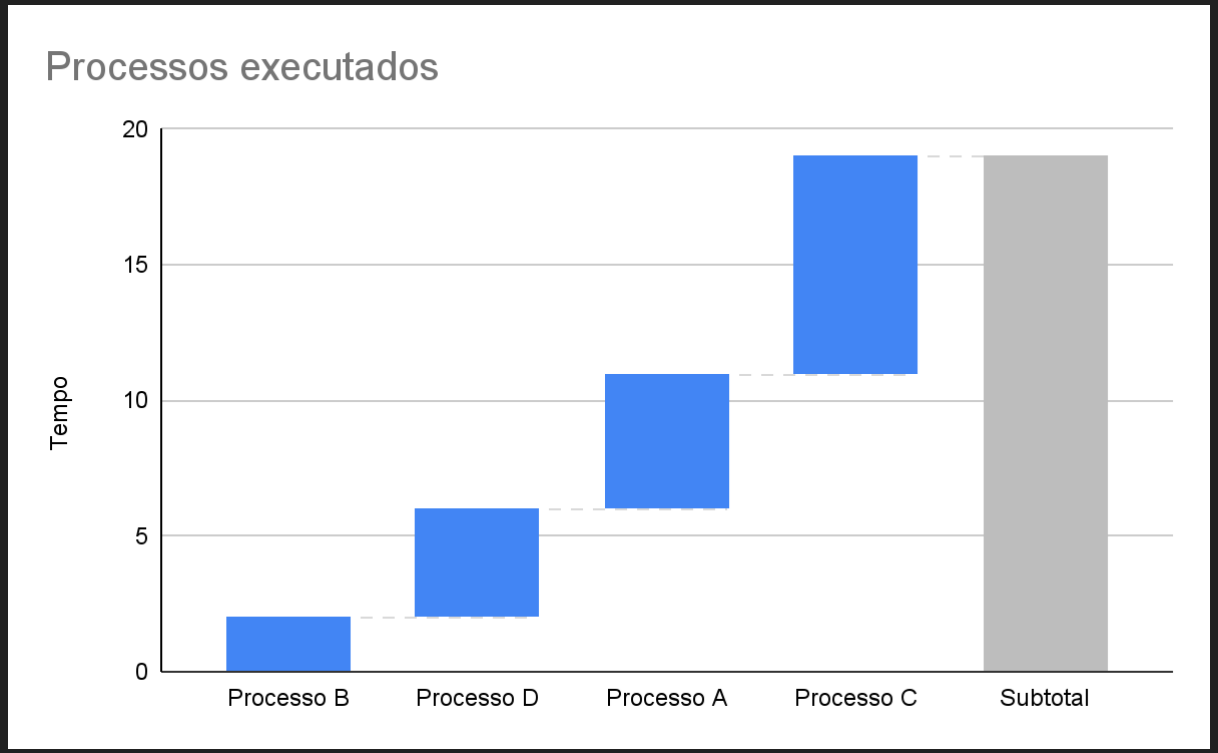
Tempos de retorno médio serão menores

Turnaround dos processos serão menores

Shortest Job First (SJF)

Processo	Tempo de processador
A	5
B	2
C	8
D	4

Shortest Job First (SJF)



Processo	Tempo de processador
B	2
D	4
A	5
C	8

Shortest Job First (SJF)

Processo	Tempo de processador
B	2
D	4
A	5
C	8

Tempo médio de retorno dos processos B, D, A e C:

(2+6+11+19) / 4 = 9.5

Shortest Job First (SJF)

```
Função simulateSJF(fila, tarefas[], numTarefas)
    Para cada tarefa de tarefas[0] até tarefas[numTarefas-1] faça
        Enfileirar(fila, tarefa)

    Enquanto o tamanho da fila for maior que zero faça
        tarefaMaisCurta <- EncontrarTarefaMaisCurta(fila)
        Se tarefaMaisCurta não for nula então
            ExecutarTarefa(tarefaMaisCurta)
            Desenfileirar(fila)
        Fim Se
    Fim Enquanto
Fim Função
```

Shortest Remaining Time Next (SRTN)

Preemptivo

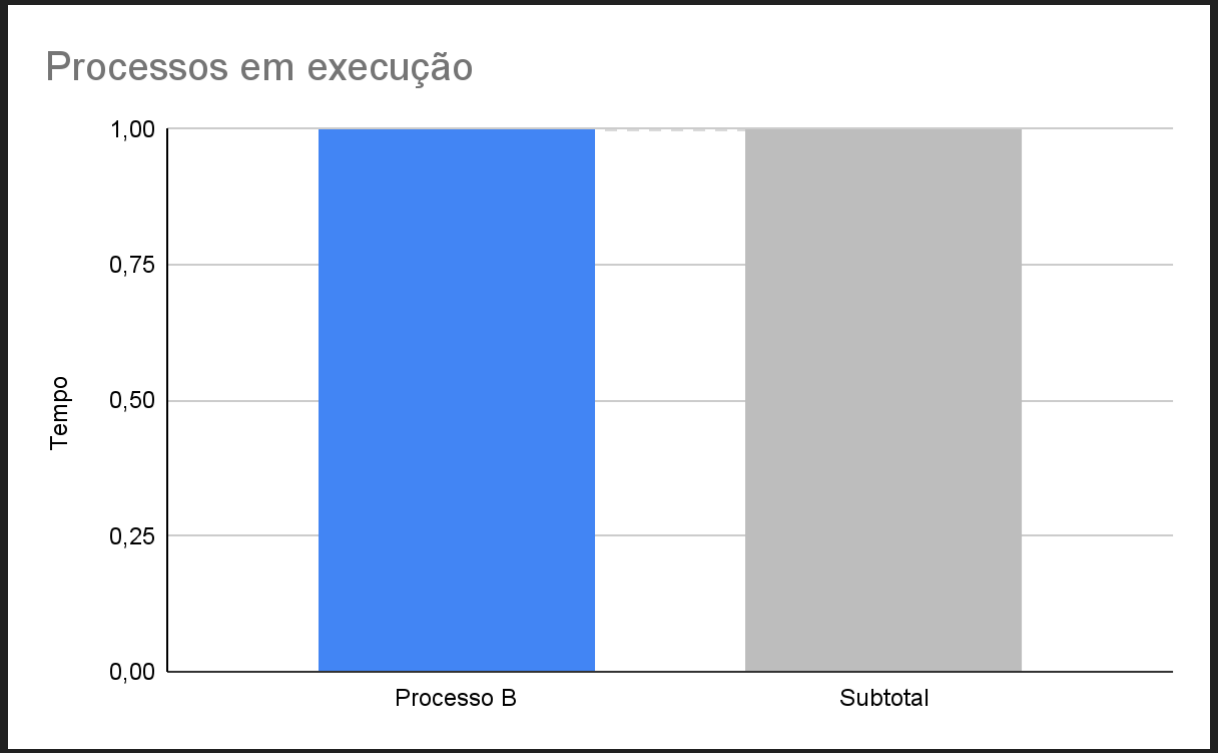
Semelhante ao SJF, pois processos com menor tempo de execução são executados primeiro

Se um novo processo com um tempo de execução menor do que o processo atual em execução aparecer, o processo em execução será interrompido para dar lugar ao de menor tempo.

Shortest Remaining Time Next (SRTN)

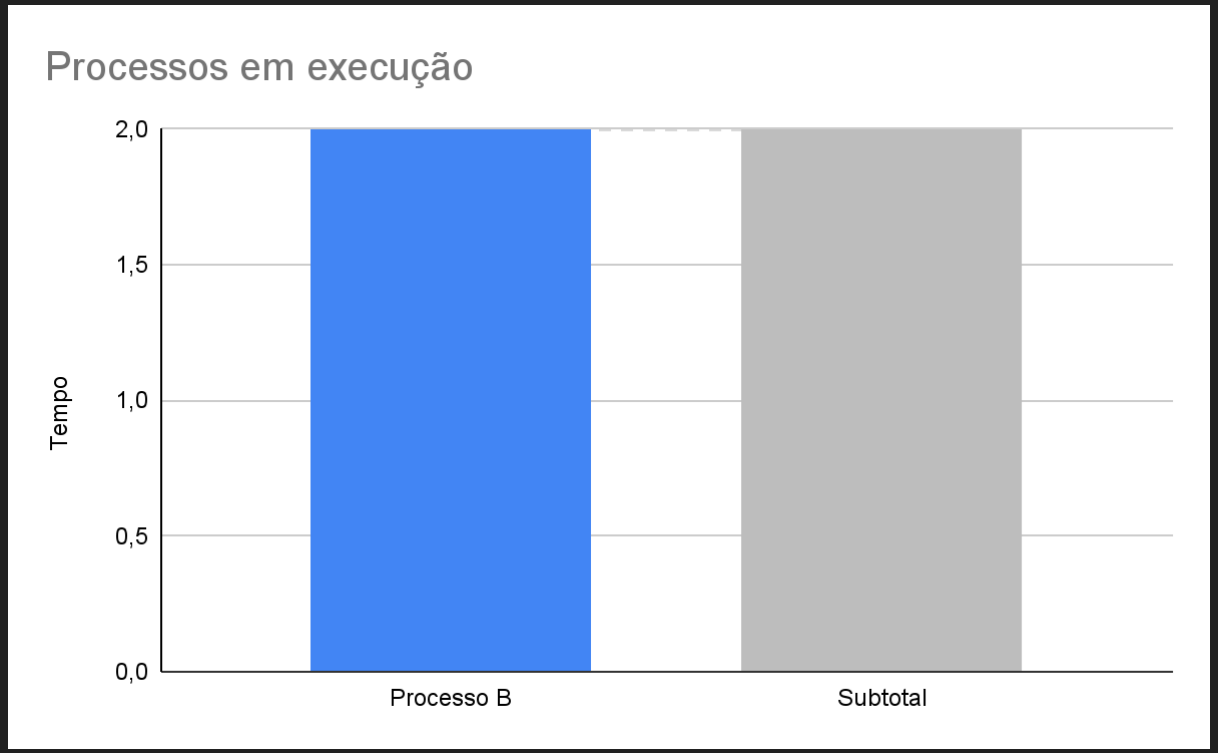
Processo	Tempo de processador
A	5
B	2
C	8
D	4

Shortest Remaining Time Next (SRTN)



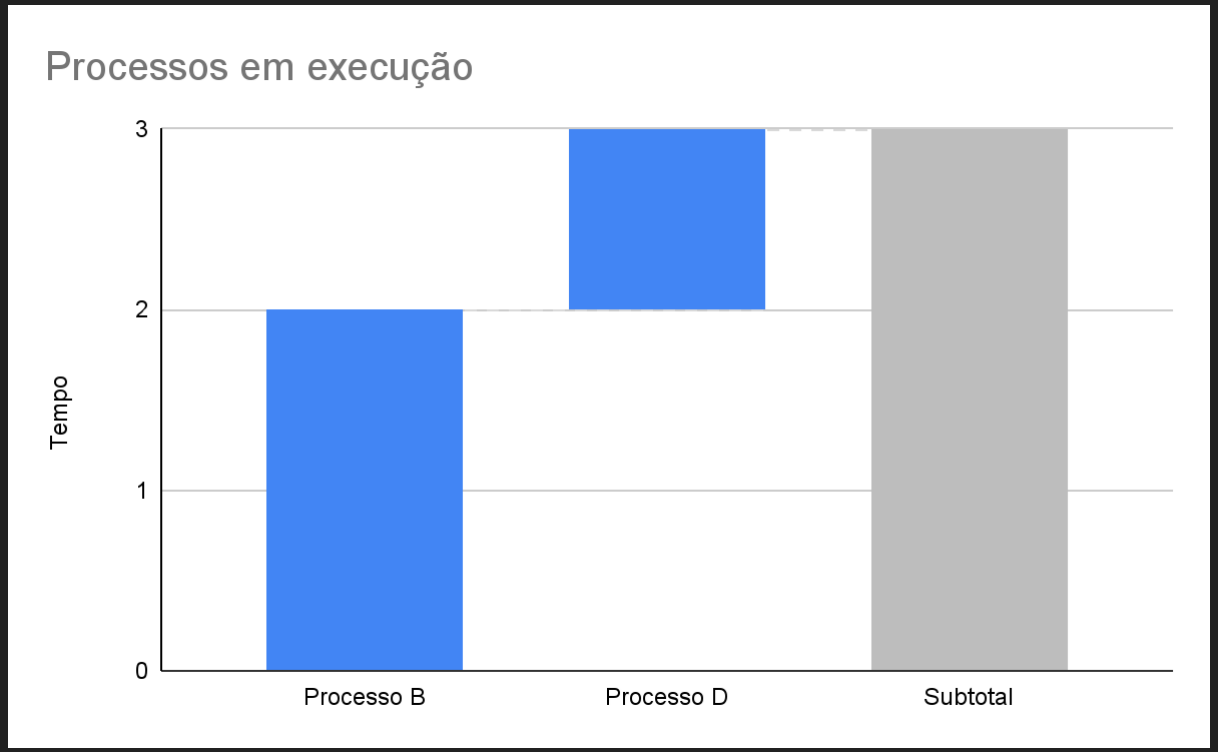
Processo	Tempo de processador
B	1
D	4
A	5
C	8

Shortest Remaining Time Next (SRTN)



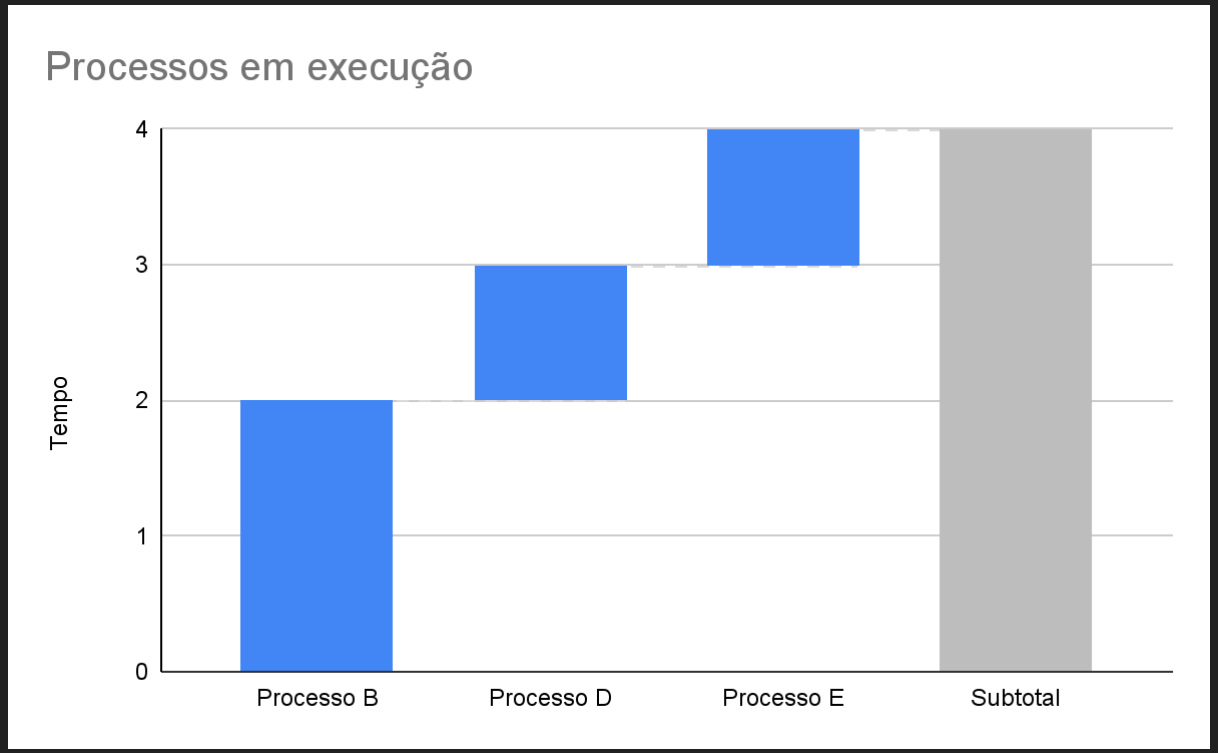
Processo	Tempo de processador
D	4
A	5
C	8

Shortest Remaining Time Next (SRTN)



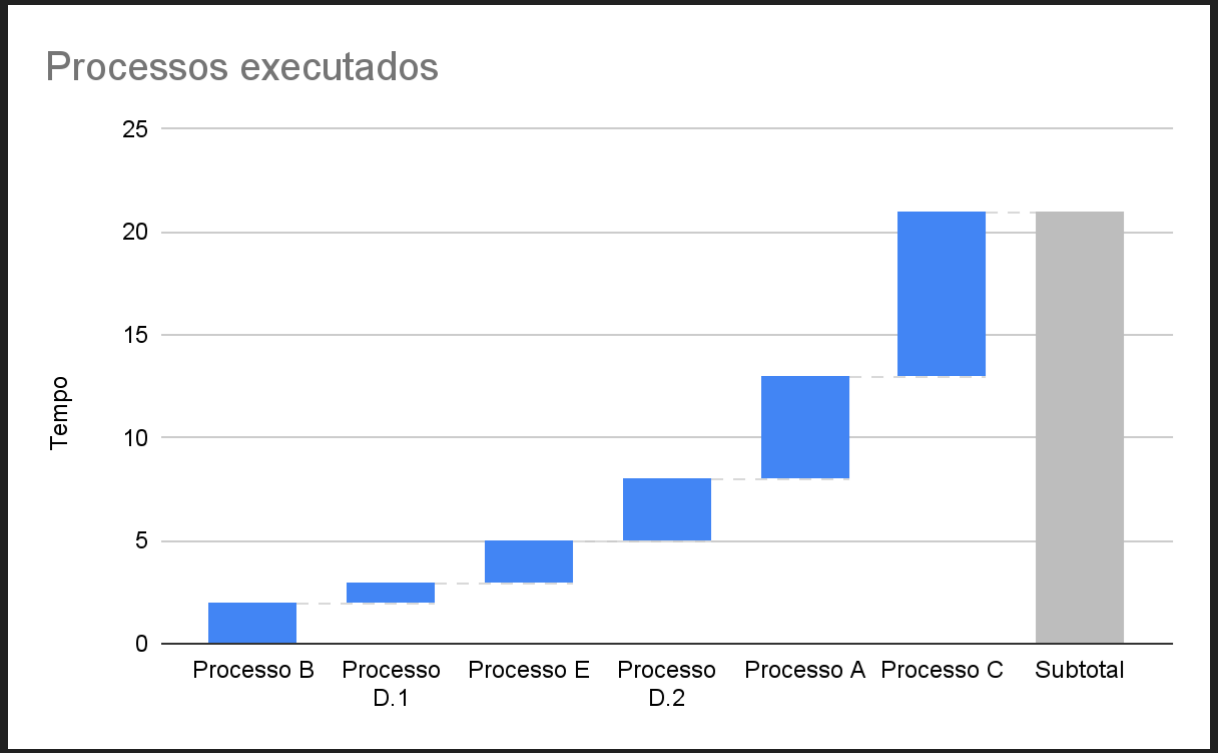
Processo	Tempo de processador
E	2
D	3
A	5
C	8

Shortest Remaining Time Next (SRTN)



Processo	Tempo de processador
E	1
D	3
A	5
C	8

Shortest Remaining Time Next (SRTN)



Processo	Tempo de processador
B	2
E	2
D	4
A	5
C	8

Shortest Remaining Time Next (SRTN)

```
Função simulateSRN(fila, tarefas[], numTarefas)
    Enquanto houver tarefas restantes ou a fila não estiver vazia faça
        tarefaMaisCurta <- NULL
        menorTempo <- INFINITO

        Para cada tarefa de tarefas[0] até tarefas[numTarefas-1] faça
            Se tarefa não estiver na fila então
                Enfileirar(fila, tarefa)

        Se a fila não estiver vazia então
            Para cada tarefa na fila faça
                Se tarefa.tempoDeExecução < menorTempo então
                    menorTempo <- tarefa.tempoDeExecução
                    tarefaMaisCurta <- tarefa
                Fim Se
            Fim Para

            Se tarefaMaisCurta não for nula então
                ExecutarTarefa(tarefaMaisCurta)
                Se tarefaMaisCurta.tempoDeExecução == 0 então
                    Desenfileirar(fila)
                Fim Se
            Fim Se
        Fim Se
    Fim Enquanto
Fim Função
```

SISTEMAS INTERATIVOS

(Filipe Correia Belfort)

Escalonador de CPU

Função : seleciona dentre os processos disponíveis na memória que estejam prontos para executar, e aloca a CPU a um deles

Decisões de escalonamento de CPU podem ocorrer quando um processo:

- Passa do estado de execução para espera
- Passa para o estado de executando para pronto
- Passa de esperando para pronto
- Encerra

Agendamento de CPU

- O agendamento de processos / trabalho é feito para terminar o trabalho no prazo
- Abaixo estão os diferentes tempos em relação a um processo

- Hora de chegada: hora em que o processo chega à fila de espera
- Tempo de Conclusão: Tempo em que o processo conclui sua execução
- Burst Time: Tempo exigido por um processo para execução da CPU

- Turn Around Time: Diferença de tempo entre a hora de conclusão e hora de chegada
- Tempo de resposta = Tempo de conclusão - Tempo de chegada
- Tempo de espera (WT): Diferença de tempo entre o tempo de rotação e o tempo de burst
- Tempo de espera = Tempo de resposta - Tempo de burst

Round Robin Scheduling

1. algoritmo de escalonamento Round Robin é preemptivo. muito útil em sistemas multitarefa
2. Cada processo é atribuído a um tempo fixo de forma cíclica
3. Cada processo recebe uma pequena unidade de tempo de CPU (quantum de tempo)
4. É projetado especialmente para sistemas de compartilhamento de tempo
5. Exemplos: escalonamento de CPU, escalonamento de pacotes, Balanceamento de carga em serviços web, agendamento de tarefas, sistemas de filas para atendimento ao cliente)
6. A fila pronta é tratada como uma fila circular
7. O processo no início da fila pronta tem a chance de ser executado primeiro, mas apenas para o período de quantum único.

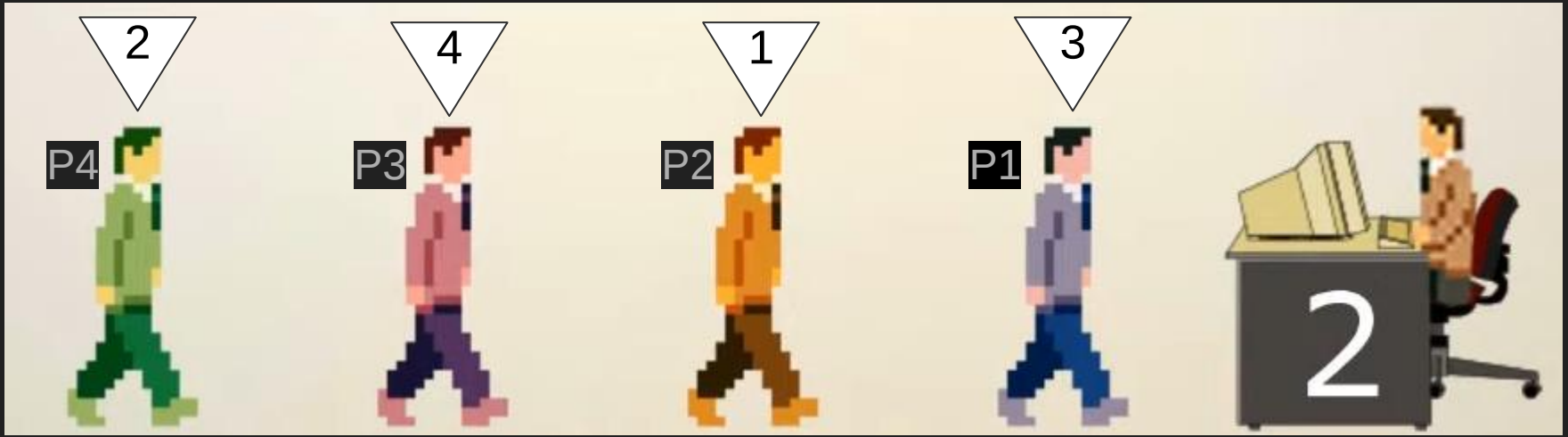
Execução do algoritmo Round Robin (1)

Número máximo de contas a pagar no caixa (quantum) = 2

Número de pessoas (processos) na fila: 4

Execução do algoritmo Round Robin (2)

Número de contas a pagar por pessoa:
pessoa 1: três
pessoa 2: uma
pessoa 3: quatro
pessoa 4: dois



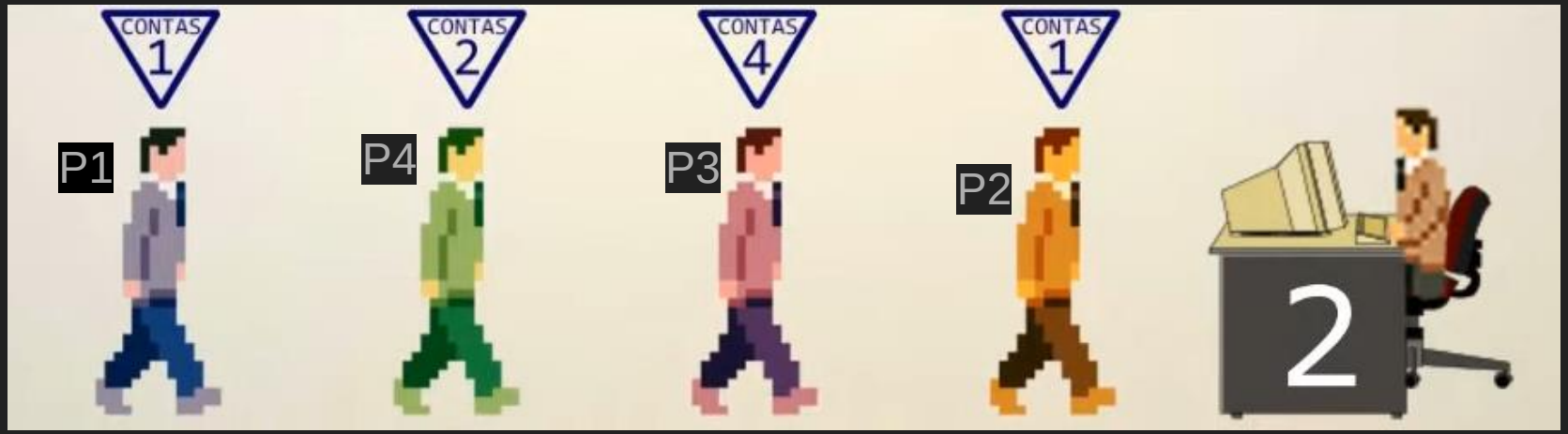
Execução do algoritmo Round Robin (3)

pessoa 1 vai para o caixa com 3 contas

paga até 2 contas.

Execução do algoritmo Round Robin (4)

pessoa 1 vai pro final da fila com uma conta restante



Execução do algoritmo Round Robin (5)

a pessoa 2 vai para o caixa com 1 conta

paga até 2 contas, e sai da fila.

Execução do algoritmo Round Robin (6)

a pessoa 3 vai para o caixa com 4 contas

paga até 2, e vai pro final da fila com 2 contas.

Execução do algoritmo Round Robin (7)

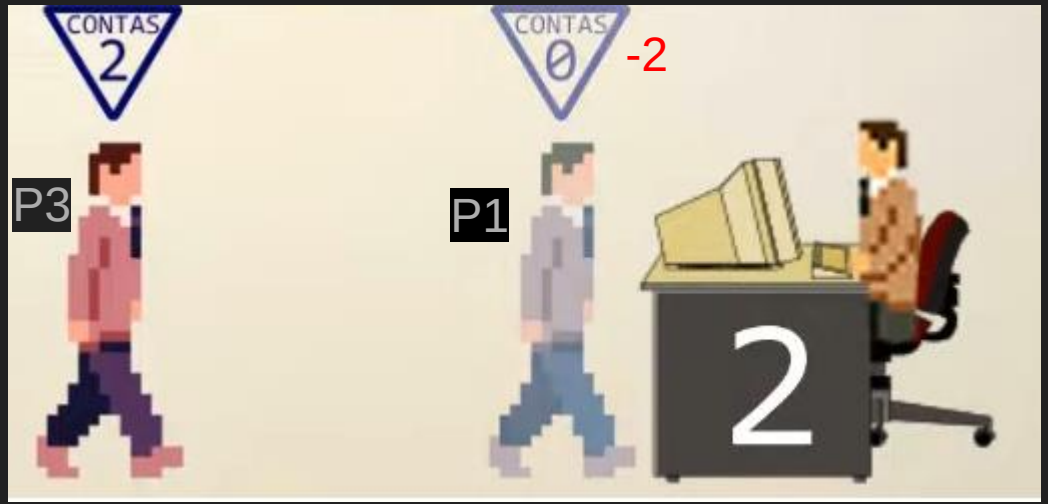
a pessoa 4 vai para o caixa com 2 contas

paga até 2 contas e sai da fila.

Execução do algoritmo Round Robin (8)

a pessoa 1 vai para o caixa com uma conta

paga até 2 e sai da fila.

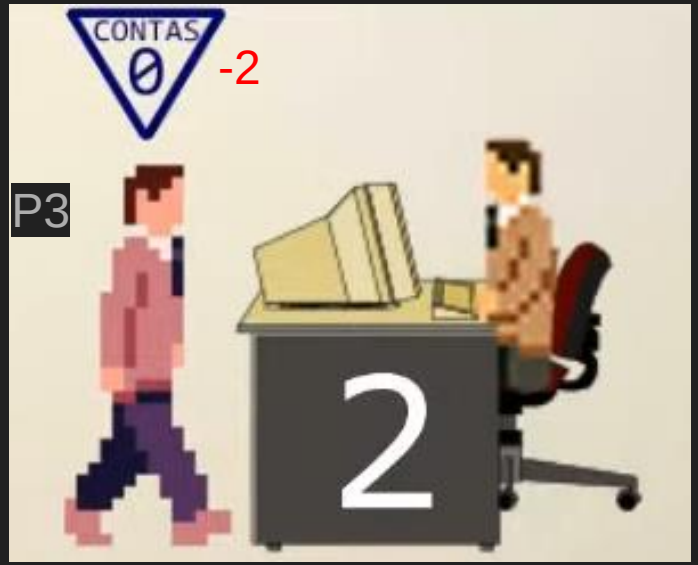


Execução do algoritmo Round Robin (9)

a pessoa 3 vai para o caixa com 2 contas

paga até 2 contas, e sai da fila.

Fim do algoritmo



Priority Scheduling

- Algoritmo não preemptivo e um dos algoritmos de agendamento mais comuns em sistemas em lote
- Se dois processos tiverem o mesmo tempo de chegada, então compare com as prioridades (processo mais alto primeiro)
- Além disso, se dois processos têm a mesma prioridade, compare com o número do processo (menos o número do processo primeiro)
- Este processo é repetido enquanto todos os processos são executados

Priority Scheduling

Os processos com a mesma prioridade são executados por ordem de chegada, A prioridade pode ser decidida com base nos requisitos de memória, requisitos de tempo ou qualquer outro requisito de recurso.

Implementação:

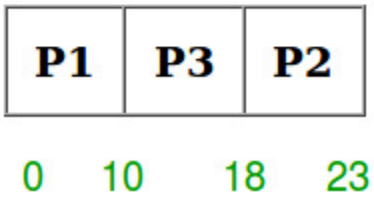
1 - Primeiro insira os processos com seu tempo de burst e prioridade

2 - Classifique os processos, tempo de burst e prioridade de acordo com a prioridade

3 - Agora basta aplicar o algoritmo FCFS (First Come First Serve)

Priority Scheduling

Process	Burst Time	Priority
P1	10	2
P2	5	0
P3	8	1

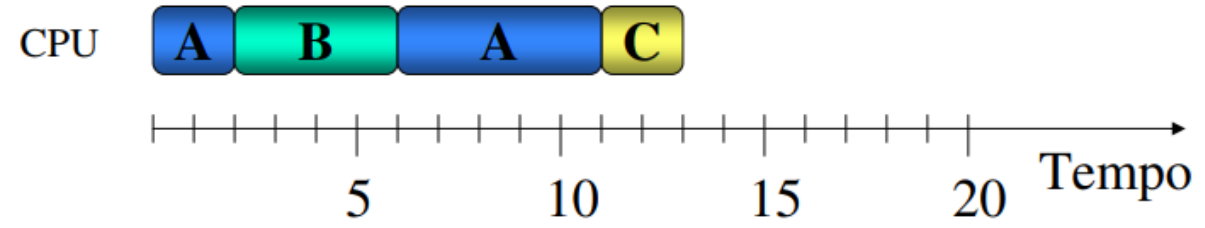


Priority Scheduling

- Um grande problema com o agendamento de prioridade é o bloqueio indefinido ou inanição
- Uma solução para o problema de bloqueio indefinido do processo de baixa prioridade é o envelhecimento
- O envelhecimento é uma técnica de aumentar gradativamente a prioridade de processos que aguardam no sistema por um longo período de tempo

Priority Scheduling

Vamos supor que os processos possuem as seguintes prioridades A=5, B=1 e C=6 (Considerando que 1 é a prioridade mais alta e 9 a mais baixa).



$$TEspera_{medio} = \frac{TEspera_A + TEspera_B + TEspera_C}{n^o\ processos} = \frac{6+0+(11-3)}{3} = \frac{6+0+8}{3} = 4{,}66$$

$$T\,\mathrm{Re}torno_{medio} = \frac{T\,\mathrm{Re}torno_A + T\,\mathrm{Re}torno_B + T\,\mathrm{Re}torno_C}{n^o\ processos} = \frac{11+6+13}{3} = 10$$

múltiplas filas

Pode acontecer que os processos na fila de espera sejam divididos em classes diferentes, onde cada classe tem suas próprias necessidades de escalonamento

uma divisão comum é um processo em primeiro plano (interativo) e um processo em segundo plano (lote) . Essas duas classes têm necessidades de agendamento diferentes.

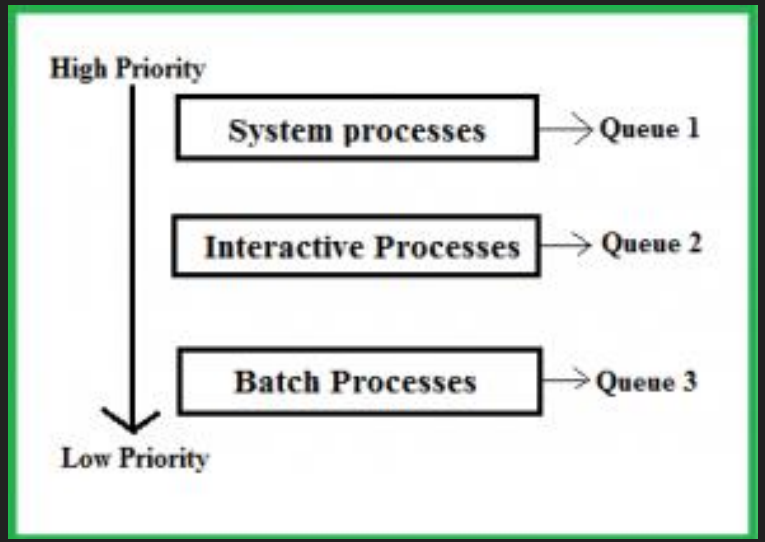
Para esse tipo de situação, o Agendamento de Fila de Vários Níveis é usado.

múltiplas filas

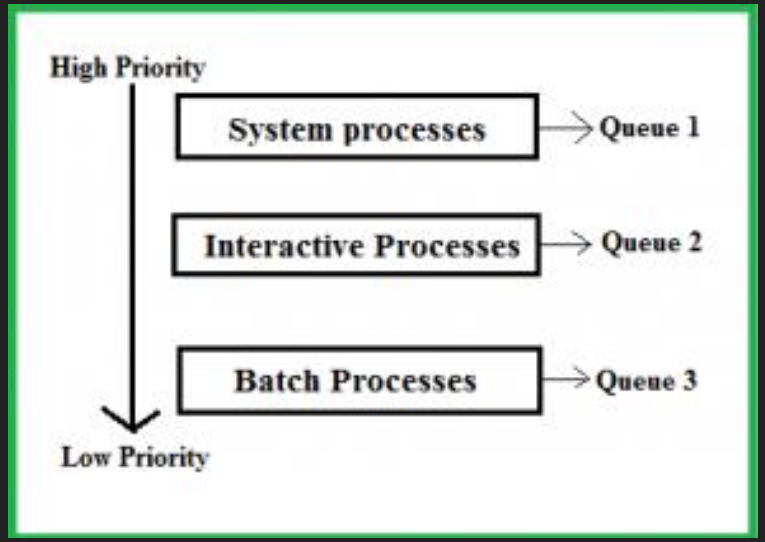
A Fila pronta é dividida em filas separadas para cada classe de processos

Por exemplo, consideremos três tipos diferentes de processos: Processos do sistema, Processos interativos e Processos em lote.

Todos os três processos têm sua própria fila. Agora, olhe para a figura abaixo.



múltiplas filas



Todos os três tipos diferentes de processos têm sua própria fila. Cada fila possui seu próprio algoritmo de agendamento. Por exemplo, a fila 1 e a fila 2 usam Round Robin, enquanto a fila 3 pode usar FCFS para agendar seus processos.

múltiplas filas

Agendamento entre as filas: O que acontecerá se todas as filas tiverem alguns processos? Qual processo deve pegar a CPU? Para determinar este Agendamento entre as filas é necessário. Existem duas maneiras de fazer isso

Método de programação preemptiva de prioridade fixa - Cada fila tem prioridade absoluta sobre a fila de prioridade mais baixa.

Exemplo: fila 1> fila 2> fila 3

Divisão de tempo - neste método, cada fila obtém uma determinada parte do tempo da CPU e pode usá-lo para agendar seus próprios processos. Por exemplo, a fila 1 ocupa 50% do tempo da CPU, a fila 2 ocupa 30% e a fila 3 ocupa 20% do tempo da CPU.

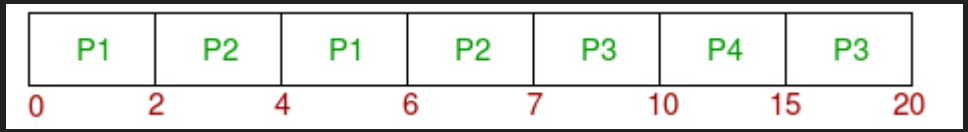
múltiplas filas

Considere a tabela abaixo de quatro processos em agendamento de fila multinível. O número da fila denota a fila do processo.

Process	Arrival Time	CPU Burst Time	Queue Number
P1	0	4	1
P2	0	3	1
P3	0	8	2
P4	10	5	1

Fila 1 usa Round Robin (Time Quantum = 2)

Fila 2 usa FCFS.



múltiplas filas

Vantagens:

- Os processos são atribuídos permanentemente à fila, por isso tem a vantagem de reduzir a sobrecarga de agendamento.

Desvantagens:

- Alguns processos podem ficar sem CPU se algumas filas de prioridade mais alta nunca ficam vazias.
- É inflexível por natureza.

SISTEMAS DE TEMPO REAL

(Filipe Correia Belfort)

Sistemas de tempo real

- Um Sistema de Tempo-Real é um sistema computacional reactivo, i.e., que reage a estímulos externos (incluindo à passagem do tempo) em intervalos de tempo impostos pelo seu ambiente (operador e objecto controlado

- Um Sistema de Tempo-Real pode ser caracterizado por ter, em geral, um funcionamento contínuo que lhe permite reagir a estímulos externos.

- A correcção de um sistema de tempo-real depende não só do resultado lógico das computações efectuadas, mas também do instante de tempo em que os resultados são produzidos [Stankovic, 1988].

- Um Sistema de Tempo-Real não é um sistema rápido, mas sim um sistema previsível

Sistemas de tempo real

Sistemas genéricos vs. Sistemas de Tempo-Real

“Executar uma determinada tarefa no menor intervalo de tempo possível” → objectivo de um “general purpose system” vs. “Nunca ultrapassar o intervalo de tempo pré-determinado para executar uma determinada tarefa” → objectivo de um STR (a consequência da perda de uma meta temporal pode ser drástica)

Sistemas de tempo real

Embedded System

É um sistema no qual a parte “física” e a parte computacional estão fortemente integradas, interagindo através de sensores e actuadores para desempenhar uma função específica, Um Sistema Embebido é também um sistema computacional reactivo, i.e., que reage a estímulos externos em intervalos de tempo impostos pelo seu ambiente;

Exemplos: “Pacemaker” e desfibrilador, Leitor de cartões inteligentes, Receptor GPS.

Sistemas de tempo real

Embedded System

Um sistema embebido é (quase) sempre um sistema de tempo-real

Contra exemplo: Um PDA que suporte uma aplicação lúdica

Nem todos os sistemas de tempo-real são sistemas embebidos

Exemplo: Sistemas de controlo de tráfego aéreo, Sistemas de negociação bolsista.

Sistemas de tempo real

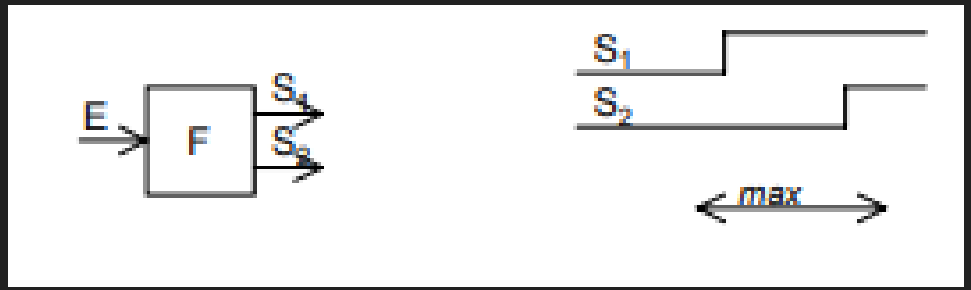
Parâmetros Temporais

Meta temporal para finalização de uma tarefa (melhor caso e pior caso)

A ultrapassagem de uma meta temporal (“deadline”) corresponde a uma avaria temporal (quando o serviço prestado não está em conformidade com a especificação)

Sincronização entre instantes de produção de resultados

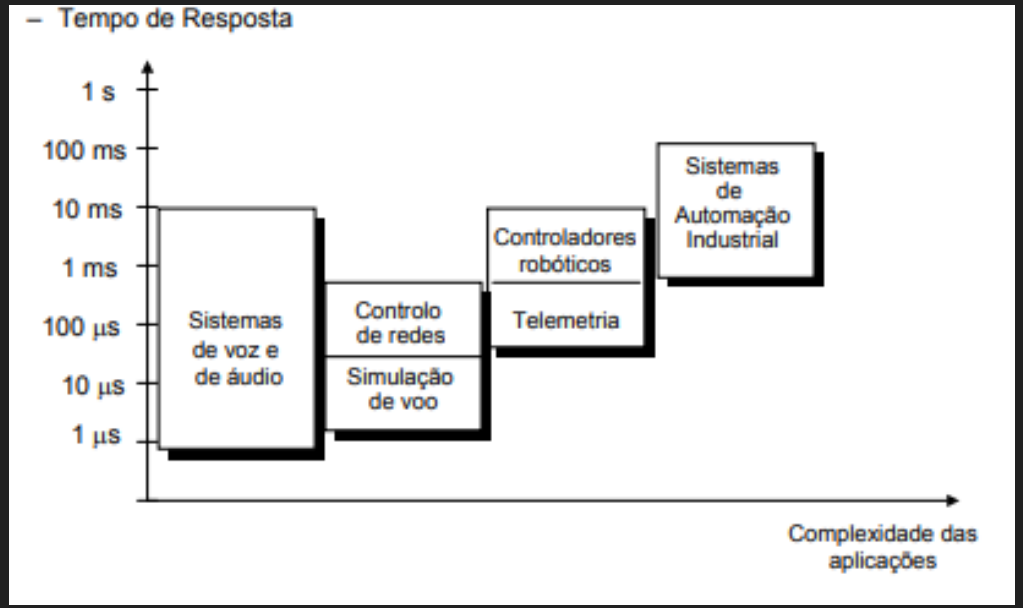
Ex. Sincronização Som – Imagem



Sistemas de tempo real

Parâmetros Temporais

Tempo de Resposta: Intervalo entre o instante em que uma ou mais entradas modificam o seu valor e o instante em que o sistema computacional reage a essas modificações, através de uma modificação do valor das suas saídas.



Sistemas de tempo real

Sistema Crítico vs. Sistema Não-Crítico

Considera-se um STR como sendo crítico caso o não cumprimento de uma meta temporal (“deadline”) tenha consequências graves para a utilidade do sistema (prejuízo causado várias ordens de grandeza superior ao benefício decorrente da sua correcta execução), de forma prática existe a necessidade de obtenção de garantias em fase de concepção;

Considera-se um STR como sendo não-crítico caso esse não cumprimento de metas temporais tenha unicamente consequências ligeiras para a utilidade do sistema; – Enquanto que num sistema crítico as metas temporais são de cumprimento “obrigatório”, num sistema não-crítico as metas temporais descrevem o comportamento temporal desejado para o sistema (por exemplo numa aplicação de videoconferência)

Sistemas de tempo real

Sistema Avaria-Seguro vs. Sistema Avaria-Operacional

Caso existam um ou mais estados seguros de funcionamento do objecto controlado, em caso de avaria o sistema computacional deve evoluir para um desses estados (“Fail-Safe”);

Caso não existam estados seguros de funcionamento, o sistema deve manter um nível mínimo de funcionalidade com segurança (“Fail-Operational System”); Nota: um estado seguro de funcionamento é função do objecto controlado e não do sistema computacional.

Sistemas de tempo real

Classificação Ortogonal

Sistemas de Resposta Garantida vs. Sistemas de Melhor Esforço – Considera-se um sistema como sendo de Resposta Garantida (“Guaranteed Response”) caso a sua concepção seja baseada na adequabilidade de recursos, ou seja, na garantia da existência de recursos computacionais suficientes para suportar os cenários máximos de carga e de falhas

Sistema determinístico, no que diz respeito à previsibilidade temporal

Considera-se um sistema como sendo de Melhor Esforço caso a sua concepção seja baseada em estratégias de alocação dinâmica de recursos, combinadas com argumentos probabilísticos acerca da simultaneidade na ocorrência de cenários máximos de carga e/ou de de falhas

Sistema probabilístico no que diz respeito à previsibilidade temporal.

Sistemas de tempo real

Classificação Ortogonal

Vantagens / Desvantagens de um Sistema de Resposta Garantida:

O cumprimento das metas temporais é garantido na fase de concepção, não havendo a necessidade de sobrecarregar o processamento “on-line” com o cálculo de testes de escalonabilidade

Vantajoso para aplicações de elevada criticidade, devido ao determinismo inerente

Requer na fase de concepção um conhecimento exacto dos pressupostos de carga e de falhas, o que poderá ser de difícil exequibilidade;

Implica uma alocação de recursos para o pior caso (carga máxima), gerando uma grande subutilização de recursos computacionais;

Sistemas de tempo real

Determinismo de execução: Previsibilidade

Deverão ser conhecidos todos os cálculos efectuados para a determinação da carga imposta sobre o sistema computacional (aplicação e sistema operativo)

Deverão ser conhecidos os tempos de execução das aplicações suportadas (no pior caso, e de uma forma não subestimada)

Só será possível caso o hardware utilizado tenha tempos de execução previsíveis.

A correcta selecção da ordem de execução das tarefas/funções é uma das funções mais relevantes num STR

Deverá ser obtida uma prova de correcto funcionamento do STR (garantia dos requisitos temporais) em tempo de concepção, a partir dos pressupostos de carga e de falhas

Referências:

Silberschatz, Galvin e Gagn, 2007, Conceitos de sistema operacional com Java – 7 a edição

TANENBAUM, A.; BOS, H. *Sistemas Operacionais Modernos*. 4ª Edição. São Paulo: Pearson Education do Brasil, 2016.

ZORZAL, E. Sistemas Operacionais.

OBRIGADO
PELA ATENÇÃO E
BONS ESTUDOS!!!

Reconhecimentos e Direitos Autorais

@autor: [Filipe Correia Belfort, Josiel Costa dos Santos Junior e Steven Roger dos Santos Soares]

@contato: [filipe.belfort@discente.ufma.br, josiel.junior@discente.ufma.br e steven.roger@discente.ufma.br]

@data última versão: [10-12-2023]

@versão: 1.0

@outros repositórios: [URLs - https://github.com/Josiel-Jr]

@Agradecimentos: Universidade Federal do Maranhão (UFMA), Professor Doutor Thales Levi Azevedo Valente, e colegas de curso.

Este material é resultado de um trabalho acadêmico para a disciplina SISTEMAS OPERACIONAIS, sobre a orientação do professor Dr. THALES LEVI AZEVEDO VALENTE, semestre letivo 2023.2, curso Engenharia da Computação, na Universidade Federal do Maranhão (UFMA). Todo o material sob esta licença é software livre: pode ser usado para fins acadêmicos e comerciais sem nenhum custo. Não há papelada, nem royalties, nem restrições de "copyleft" do tipo GNU. Ele é licenciado sob os termos da licença MIT reproduzida abaixo e, portanto, é compatível com GPL e também se qualifica como software de código aberto. É de domínio público. Os detalhes legais estão abaixo. O espírito desta licença é que você é livre para usar este material para qualquer finalidade, sem nenhum custo. O único requisito é que, se você usá-los, nos dê crédito.