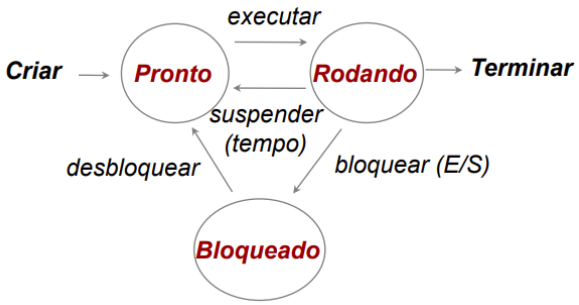
- **Sistema operacional =** É um gerenciador de recursos do computador que fornece uma interface amigável com o usuário.
- Ao ligarmos o computador:
    - 1. A CPU vai ao endereço inicial da memória principal (onde está o BIOS [Basic Input-Output System]) e executa o POST (Power On Self Test), que testa todos os periféricos.
    - 2. A CPU executa o Boot Loader, que vai ao setor 0 (zero) do disco buscar o Loader do S.O e o carrega na memória principal.
    - 3. A CPU executa o Loader do S.O. que carrega o S.O. na memória principal e passa a executá-lo.
        - Caso exista mais de uma opção de S.O., no passo 2 é carregado o software de seleção e o usuário faz sua escolha antes do passo 3.
- **Processo =** um programa em execução.
    - Criação de Processos: Início do sistema, Execução de chamada ao sistema de criação de processos, Solicitação do usuário para criar um novo processo.
    - Término de Processos: Saída normal (voluntária), Saída por erro (voluntária), Erro fatal (involuntário), Cancelamento por um outro processo (involuntário).
- **Espaço de Endereçamento =** área de memória alocada a um processo.
- **Estados de um processo =**
    - Rodando: O processo está em execução pela CPU
    - Pronto: O processo está aguardando que o escalonador o coloque em execução
    - Bloqueado: O processo está esperando o término de algum evento de Entrada/Saida para ir para o estado Pronto.




    - *O que ocorre quando não há nenhum processo do usuário no estado "Rodando"?* O Escalonador põe o Processo Ocioso do Sistema para rodar.
    - *O que faz com que um processo deixe o estado "Rodando" direto para o estado "Pronto"?* Uma interrupção do temporizador.
    - *Em que estado(s) do escalonamento é possível não ter nenhum processo? Por quê?* No estado Bloqueado, caso nenhum processo esteja esperando por Entrada/Saída, e no Pronto, quando todos estão bloqueados esperando por Entrada/Saída.
    - *O que acontece com o processo que está sendo executado quando ocorre uma interrupção ou uma exceção? O processador para a execução e lê uma parte específica do código chamada Tratador de Interrupção, e após executá-la geralmente o programa volta a ser executado. (resposta por Marçal may not be correct)*
    - *Como um computador com um único processador faz para executar vários programas ao mesmo tempo? Explique como isso é feito.  Graças ao*

*escalonador de processos, diferentes programas são executados "ao mesmo tempo" graças ao time-slice, ou seja, a partição do tempo de execução em tempos muito pequenos, que fazem com que vários programas tenham respostas quase que simultaneamente, assim não é preciso fechar um para ter a resposta em outro. (resposta por Marçal may not be correct)*

- **Contexto de um processo =** conjunto de informações necessárias para retomar a execução a partir do ponto em que este parou. O Contexto contém contador de programa, registradores, estado, pilha, espaço de endereçamento, arquivos abertos, processos filhos, informações de uso do contabilidade e uso do sistema.
- **Proteção de memória de processos diferentes =** manutenção da integridade dos dados de um processo na presença de outros processos. Implementada em conjunto pelo hardware e pelo sistema operacional.
- **Página =** parte de um programa capaz de caber na memória
- **Memória virtual =** espaço de armazenamento de páginas em disco
- **Interrupção =** forma de um dispositivo de E/S chamar a atenção da CPU / S.O.. É algo desejado e que facilita a comunicação com os dispositivos (evita, por exemplo, o “busy-waiting”).
    - Para obter entrada e saída de dados, não é interessante que a CPU tenha que ficar continuamente monitorando o status de dispositivos como discos ou teclados. Num sistema simples, CPU deve esperar a execução do comando de E/S. Um sistema com interrupção não fica esperando.
    - Por hardware (assíncrona)
        - Algum dispositivo externo à CPU (ex. teclado)
        - Ocorrem independentemente das instruções que a CPU está executando
        - Temporizador (para suspender um processo)
    - Por software (trap, síncronas)
        - Traps são instruções de programas que fazem com que o processador aja de forma similar ao da ocorrência de uma interrupção causada por um periférico, passando a executar rotinas em endereços pré-determinados.
        - São usadas para que programas do usuários acionem rotinas do Sistema Operacional. Por exemplo, para requisitar um serviço de entrada ou saída de algum periférico.
            - Um programa de usuário não pode chamar diretamente uma rotina do sistema operacional, já que o SO tem o seu próprio espaço de endereçamento.
        - Execução de instrução de programa
        - Situações em que o programa não teria como prosseguir (ex. overflow em operações aritméticas)
            - Como as interrupções síncronas ocorrem em função da instrução que está sendo executada, nesse caso o programa passa algum parâmetro para o tratador
- **Exceção =** algo indesejado provocado por algum erro na execução de um processo. É gerada dentro da CPU e pode ocasionar erros fatais, que encerram a execução de um processo de forma abrupta.
    - Alguns tipos de exceção: overflow, underflow, divisão por zero, instrução inválida (código da operação inválido), violação de acesso (tentativa de

acesso a regiões de memória não permitidas). A CPU passa a executar a rotina do S.O. adequada para o tratamento desta exceção.

- **Escalonador =** parte do S.O. responsável por definir a ordem de execução dos processos baseado no algoritmo de escalonamento. Quando um ou mais processos estão prontos para serem executados, o sistema operacional deve decidir qual deles vai ser executado primeiro, e isso é responsabilidade do escalonador.
    - Características:
        - Justiça (fairness) – Todos os processos têm chances iguais de uso dos processador
        - Eficiência – Taxa de ocupação do processador ao longo do tempo
        - Tempo de Resposta – Tempo entre a ocorrência de um evento e o término da ação correspondente
        - Turnaround – “Tempo de resposta” para usuários em batch
        - Throughput – Núm de “jobs” (processos) executados por unidade de tempo
    - Tipos:
        - Escalonamento Preemptivo
            - Permite a suspensão temporária de processos
            - Quantum ou time-slice: período de tempo durante o qual um processo usa o processador a cada vez
                - Fatia grande: Menos mudanças de contexto e overhead do S.O.
                - Fatia pequena: Bom para processos interativos, mais tempo gasto com a mudança de contexto
        - Escalonamento Round-Robin
            - Preemptivo, bom para sistemas interativos
            - Uso de uma lista de processos sem prioridade, simples e justo
        - Escalonamento First-In First-Out (FIFO)
            - Não-preemptivo, bom para sistemas em batch
            - Uso de uma lista de processos sem prioridade, simples e justo
- **Temporizador =** mecanismo que causa uma interrupção ao término do intervalo de tempo designado àquele processo pelo escalonador, para que um processo não execute tempo demais.
    - Antes de colocar um processo em Rodando, o escalonador ajusta o temporizador para interromper a execução daquele processo caso passe o tempo limite permitido para a sua execução. Se essa interrupção ocorrer, a CPU põe o escalonador para escolher o próximo processo para executar.
    - A cada interrupção do Temporizador, o Escalonador é executado. O Escalonador põe o processo em estado Pronto e escolhe um processo para colocar em execução, com base na Política de Escalonamento.
- **Threads =** são partes de um processo que são escalonadas individualmente. Segmento de código de um processo, que compartilham o mesmo espaço de endereçamento e podem ser escalonadas de forma independente (executam em paralelo). Em alguns sistemas, um processo só pode ter uma thread.
    - Pra quê servem? Programas que precisam de mais poder computacional em sistemas com vários processadores usam threads. Dão mais simplicidade para programar tarefas essencialmente paralelas.

    - As partes do contexto de uma thread que são compartilhadas com o processo são: espaço de endereçamento, arquivos abertos, processos filhos, informações de uso do contabilidade e uso do sistema. Não são compartilhados: contador de programa, registradores, estado e pilha.
- **Multiprocessamento =** O índice do processo contém o apontador para a lista de processos. Uma troca de processos consiste em trocar o valor dos registradores de contexto da CPU
    - O que é necessário para haver multiprocessamento?
    - Suporte do Hardware
        - Temporizadores (timers ) – Interrupções – Gerenciamento de memória – Proteção de memória
    - Suporte do S.O.
        - Escalonamento dos processos – Alocação de memória – Gerenciamento dos periféricos
- **Driver de dispositivo =** um software que é parte do S.O. e é responsável por fazer a comunicação entre a Camada Independente de Dispositivo (ou Subsistema de E/S) e o Controlador de Dispositivo. Ele traduz comandos de alto nível (usuário) para o controlador, corrige e informa erros, e pode ser instalado a qualquer momento. Eles “escondem” do subsistema de E/S (do S.O.) as diferenças entre os diversos controladores, fornecendo uma interface padrão e uniforme de acesso para todos.
    - Para instalar um dispositivo que não era conhecido quando o sistema operacional foi projetado, o usuário deve instalar um driver para que o dispositivo funcione corretamente.
    - Facilitam o desenvolvimento do S.O.
    - Permitem a inclusão de novos dispositivos
    - Detecção e correção de erros
- **Controlador de dispositivo =** são hardwares que controlam portas, barramentos e dispositivos, executando comandos de E/S.
    - Ex: Controlador da porta serial Controlador SCSI (Small Computer-Systems Interface) Controlador de disco
    - Comunicação S.O.(CPU) - Controlador
        - Controlador tem registradores de dados, comandos, status.
        - CPU pode acessar info desses registradores
        - Controlador processa comando
        - Resultados (se houver) são gravados na memória
- **Barramentos (bus) =** conjunto de condutores elétricos e com um protocolo mais complexo do que as portas, permite a comunicação entre vários componentes
    - Dispositivos:
        - Ativos ou Mestres - dispositivos que controlam o protocolo de acesso ao barramento para leitura ou escrita de dados
        - Passivos ou Escravos - dispositivos que simplesmente obedecem a requisição do mestre.
    - Exemplo: CPU ordena que o controlador de disco leia ou escreva um bloco de dados. A CPU é o mestre e o controlador de disco é o escravo.
- **Portas (ports) =** faz comunicação ponto a ponto (entre dois dispositivos), protocolo simples de comunicação.
    - Ex: Porta serial e paralela

- **Spooling e reserva de dispositivo =** um spool é um buffer que guarda as saídas a serem enviadas a um dispositivo (ex. Impressora) que não pode aceitar dados misturados de vários processos.

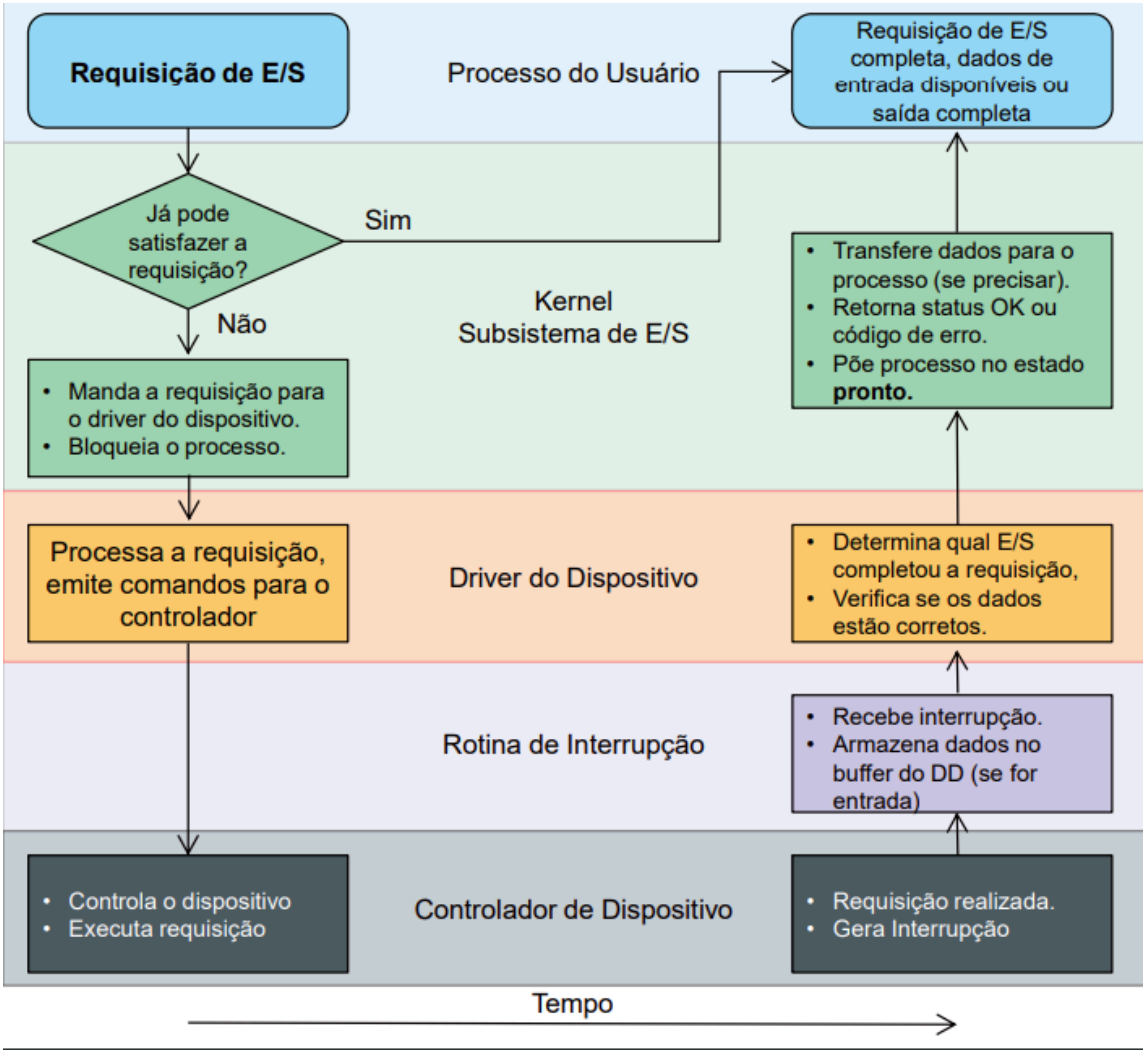
Como a CPU acessa a informação?

- **E/S isolada =** Na E/S Isolada a forma de acesso aos dispositivos de E/S é diferente do acesso à memória. Necessita de instruções especiais que apenas o S.O. pode executar, impedindo que processos do usuário tenham acesso direto aos dispositivos de E/S, promovendo maior segurança ao sistema.
- **E/S mapeada em memória =** Na E/S Mapeada em Memória a forma de acesso aos dispositivos e à memória é igual. Tem a vantagem de ter uma implementação mais simples e barata, mas não garante a proteção da E/S Isolada.

Como a CPU sabe que o dispositivo já executou o comando?

- **E/S programada =** Na E/S Programada um programa fica em loop perguntando ao dispositivo se pode enviar/receber o próximo dado até que tenha enviado/recebido todos. É extremamente ineficiente.
- **E/S por interrupção =** Na E/S por Interrupção, o S.O. envia/recebe um caracter do dispositivo e passa a executar outras atividades. Quando o dispositivo estiver pronto para outra comunicação, este gera uma interrupção fazendo com que o S.O. envie/receba o próximo caracter. Isto continua até que todos os caracteres tenham sido transmitidos.
- **E/S por DMS (Acesso Direto à Memória) =** Na E/S por DMA, um Controlador de DMA é programado para fazer toda a transferência de dados entre o dispositivo e a memória. Quando terminar, o Controlador de DMA gera uma interrupção que informa ao S.O. que terminou.

**Diga o que acontece quando um processo faz uma chamada de E/S. Mostre usando as camadas do Sistema Operacional.**




A Interrupção Passo a Passo

- Passo 1: O periférico envia o sinal de interrupção;
- Passo 2: A CPU reconhece o sinal de interrupção e salva na pilha o valor contido no Contador de Programa;
- Passo 3: A CPU identifica quem gerou a interrupção e carrega no Contador de Programa o endereço da rotina de tratamento correspondente (Vetor de Interrupção);
- Passo 4: A CPU executa a rotina de tratamento de interrupção;
- Passo 5: A CPU desempilha o valor antigo do Contador de Programa e volta a executar do ponto onde estava antes da interrupção.

Interrupção do Temporizador (Timer)

- Passo 1: O Temporizador envia o sinal de interrupção;
- Passo 2: A CPU reconhece o sinal de interrupção e salva na pilha o valor contido no Contador de Programa (ponteiro para o processo atual);
- Passo 3: A CPU identifica que a interrupção é do Temporizador e carrega no Contador de Programa o endereço do Escalonador;

- Passo 4: O Escalonador salva na memória o Contexto do processo que estava executando;
- Passo 5: O Escalonador decide qual processo é o próximo a executar, recupera da memória o Contexto deste processo e escreve nos registradores da CPU;
- Passo 6: O Escalonador recupera da memória o valor do Contador de Programa do novo processo e salva na pilha;
- Passo 7: A CPU executa a instrução de Retorno de Interrupção que remove da pilha o valor armazenado no Passo 6 e o armazena no Contador de Programa, fazendo com que a CPU passe a executar o processo escolhido no Passo 5

E/S Por Interrupção

- Passo 1: O periférico envia o sinal de interrupção;
- Passo 2: CPU salva na pilha o valor contido no Contador de Programa para poder voltar ao ponto de execução em que estava antes da interrupção;
- Passo 3: A CPU identifica quem gerou a interrupção e carrega no Contador de Programa o endereço da rotina de tratamento correspondente (Vetor de Interrupção);
- Passo 4: A CPU executa a rotina de tratamento de interrupção (ISR – Interrupt Service Routine). No início desta rotina, todos os registradores da CPU são salvos na pilha. Ao final da execução, o valor destes registradores são restaurados.
- Passo 5: A CPU reconhece (acknowledgment) o sinal de interrupção, fazendo com que o periférico desative o seu sinal de interrupção (evita entrar novamente na rotina de tratamento de interrupção).
- Passo 6: A CPU desempilha o valor antigo do Contador de Programa e volta a executar do ponto onde estava antes da interrupção.