**Atividade de Avaliação 1**

**1) Explique o que é um Sistema Operacional (S.O) e por que precisamos dos mesmos? (1.0)**

A definição do que é sistema operacional não poderia ser resumida em poucas palavras sem deixar de faltar algumas informações. Em poucas palavras, poderia se dizer que um sistema operacional consiste de um software principal que consegue gerenciar outros softwares de um computador, e também fazer a integração com o hardware do mesmo, mas a utilização de um SO é muito mais abrangente e diversa. Um sistema operacional controla quais processos são executados e os aloca entre diferentes CPUs, se você tiver um computador com várias CPUs, permitindo que vários processos sejam executados em paralelo. Além disso, ele também gerencia a memória interna do sistema, alocando memória entre aplicativos em execução.

A necessidade humana dos sistemas operacionais na atualidade vai além do seu uso em um computador pessoal, como por exemplo em inúmeros sistemas embarcados. Quando vemos dispositivos inteligentes como celulares, tablets, smartwatches, smartTvs, painéis automotivos, ou até mesmo geladeiras e microondas, todos esses dispositivos utilizam-se de sistemas operacionais para seu funcionamento, ou seja, o dia-a-dia da humanidade inteira atualmente depende de sistemas operacionais.
Os mais conhecidos na atualidade são os sistemas para computadores MacOS, Windows, distribuições baseadas em Linux e ChromeOS. Já nos smartphones, os mais populares são Android e IOS.

**2) Qual é a diferença entre modo núcleo e modo usuário? Explique como ter dois modos distintos ajuda no projeto de um sistema operacional. (1.5)**

No modo usuário, os programas que estão sendo executados não possuem acesso direto a endereços de memória e nem a permissões de alterações de hardware, dessa forma garantindo que nenhuma parte da memória será violada por outro aplicativo e impedindo danos irreversíveis ao sistema. A maior parte dos comandos executados são no modo usuário.
Já o modo núcleo (Kernel), tem acesso a partes da memória em que pode escrever, acessar e ler. Esse modo é utilizado para instruções em nível mais baixo, onde há alta confiabilidade de que o sistema irá se manter em pleno funcionamento.

Como fica evidente, a serventia de haver dois modos de acesso garante a integrabilidade e funcionamento correto do sistema, impedindo que o usuário acesse áreas restritas que poderiam corromper e inutilizar o SO.

**3) Explique o que são chamadas ao sistema. Apresente 3 exemplos de syscalls, informando se elas são POSIX ou não, e explique sua funcionalidade. (1.5)**

Chamadas ao sistema, basicamente, são funções onde é solicitado ao sistema executar determinado comando. Esses comandos podem ser relacionados a comunicação, criação e/ou execução de processos ou serviços do hardware.

Exemplos:

(UNIX) **fork**() = A chamada do sistema fork() é usado para criar processos. Não precisa de argumentos e retorna uma ID de processo. O objetivo do fork é criar um novo processo, que se torne filho do chamador. Após a criação de um novo processo filho, ambos os processos executarão a próxima instrução após a chamada do sistema fork. Portanto, temos que distinguir o pai do filho. Isso pode ser feito testando o valor devolvido do fork( ).

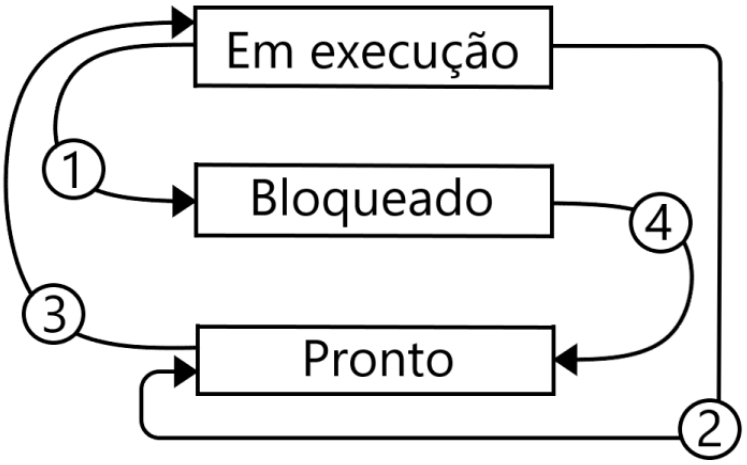
(POSIX) **mkdir** = Cria um novo diretório. O diretório é criado com as permissões de acesso especificadas e restritas pelo processo de chamada.

(Windows) **ExitProcess()** = Esta chamada do sistema é usada para sair de um processo e com isso, finalizar todos seus threads. Sair de um processo não faz com que os processos filhos sejam encerrados.

**4) Explique os estados possíveis de processos e as transições entre esses estados. Faça uma figura que ajude na explicação. (1.5)**

Três tipos de estados:

- Execução (running) – O processo está sendo executado pela CPU.
- Pronto (ready) – O processo está pronto e esperando para ser executado pela CPU.
- Bloqueado/Espera (wait) – O processo está esperando algum evento externo ou por algum recurso para poder prosseguir seu processamento.




Transições:

1. O processo passa para bloqueado quando aguarda a conclusão de um evento solicitado.
2. O processo passa de execução para pronto quando o escalonador seleciona outro processo.
3. Quando um processo é criado, é colocado em uma lista de processos no estado pronto. Então é escolhido pelo escalonador para ser executado.
4. O processo passa para pronto quando a operação solicitada é atendida ou o recurso esperado é concedido.

**5) Explique a diferença entre processos e threads. (1.0)**

Basicamente, threads são fluxo de execução sequencial dentro de um processo. Dentro de um processo, as threads compartilham o espaço de endereçamento e o contexto de software, porém cada thread possui seu contexto de hardware individual.
A comunicação entre as threads é feita de forma mais simples e mais eficiente e mais rápida. As threads de um mesmo processo podem compartilhar recursos como arquivos, temporizadores, sinais, atributos de segurança, etc. Já os processos, possuem individualmente seus espaços de endereçamento, variáveis globais, arquivos abertos, processos filhos, alarmes pendentes, sinais e tratadores de sinais e informações de contabilidade.

**6) Qual a vantagem de termos uma aplicação multithread em relação a uma aplicação monothread? Explique um exemplo de como uma aplicação poderia dividir tarefas entre threads. (1.5)**

Em um ambiente monothread um processo suporta apenas um único programa em seu espaço de endereçamento. Neste ambiente os sistemas concorrentes são implementados com o uso de múltiplos processos independentes ou subprocessos.
Com o conceito de múltiplos processos cada funcionalidade do software em execução implica na criação de um novo processo. O problema é que quanto mais processos em execução pelo sistema operacional mais recursos são consumidos por estes processos. Sempre que um novo processo é criado, recursos são alocados a ele. Outro problema é que como os processos são independentes, cada um tem seu próprio espaço de endereçamento e por isso a comunicação entre os processos é mais difícil de ser feita e mais lenta, pois utiliza mecanismos tais como troca de mensagem ou memória compartilhada. Em ambientes multithread cada processo pode responder a várias solicitações concorrentemente ou mesmo simultaneamente, no caso de haver mais de um processador. A vantagem no uso de threads é minimizar a alocação de recursos com a criação de novos processos.

Exemplo: Ao abrirmos um navegador, estamos criando um processo para executar aquele programa. Dentro do navegador nós podemos abrir várias guias, fazer um download em uma, enquanto escutamos música em outra e também enquanto digitamos um texto num documento online. Tudo isso é possível com o uso de multithreads desempenhando várias funções dentro do mesmo processo.

**7) Julgue como Verdadeiro ou Falsas as sentenças abaixo. Para as sentenças avaliadas como Falsas, justifique sua resposta. (2.0)**

A. **O OS/360 foi um dos primeiros SOs significativos na história e tinha por característica ter apenas alguns milhares de linhas de código e ser um sistema que apresentou o conceito de máquinas virtuais.**

**B. O espaço de endereçamento é o conjunto de endereços que um programa de usuário pode utilizar para identificar a localização das chamadas ao sistema ofertadas pelo SO.**

**C. O conceito de time-sharing se refere ao fato de termos diferentes processos ocupando simultaneamente áreas diferentes da memória em um determinado sistema computacional.**

**D. Um thread pop-up é um thread que é criado por um processo para lidar com uma nova mensagem que foi recebida pelo mesmo. Por exemplo, imagine um processo do WhatsApp que ao receber uma mensagem, um novo vídeo de um amigo, cria uma thread para lidar com o download deste vídeo e depois gerar uma notificação para o usuário.**

**E. O uso de threads é mais seguro que o uso de processos, já que o sistema operacional implementa soluções para controlar que o funcionamento de uma thread não interfira em outra.**

A. Falso. Tinha por característica ter milhões de linhas de código.

B. Falso. Espaço de endereçamento seria a área da memória do processo onde o programa será executado e para dados utilizados por ele (Código, dados, pilha, etc.).

C. Verdadeiro.

D. Verdadeiro. A nova thread seria ativada por meio de uma requisição. Ela atenderá o serviço e se encerrará em seguida.

E. Falso. Uso de threads é perigoso. Um thread pode fazer qualquer coisa dentro do espaço de endereçamento. Pode ler, escrever e até apagar a pilha de outro thread.