Guilherme Campos Santos

2°semestre Ciência da computação

UniCEUB - Taguatinga Campus II

Lista-1

1-O Hardware fornece os recursos computacionais para o sistema operacional que controla as aplicações sobre o manuseio do homem.

2-São sistemas onde um programa principal invoca a rotina do serviço, um conjunto de rotinas de serviço executam as chamadas de sistema e um conjunto de rotinas utilitárias que auxiliam as rotinas de serviço.

O sistema operacional é executado como um único programa, escrito como uma coleção de procedimentos com parâmetros e resultados que podem chamar umas às outras.

Tem como principal vantagem o tempo de resposta das tarefas.

3-São sistemas com hierarquia de serviço entre as camadas.

Camadas: Operador;

Programa usuário;

Gerenciamento de entrada e saída;

Comunicação operador-processo;

Gerenciamento de memória;

Alocação do processador e multiprogramação.

As camadas superiores prestam serviço aos inferiores e cada camada é dependente.

4- O sistema operacional é dividido em partes com cada uma responsável por um tipo de serviço, é dividido em: comunicação entre processos, gerenciamento de memória e escalonamento de processos. A comunicação acontece por troca de mensagens entre os programas e os serviços.

5-No Sistema Monolítico a arquitetura de kernel oferece a vantagem dos procedimentos com parâmetros e resultados que podem chamar umas as outras e diminuir o tempo de resposta das tarefas.

No Microkernel deu a opção de separar o sistema operacional em partes sendo cada uma responsável por um tipo de serviço e também permite a troca de mensagens entre ambos.

No sistema em camadas surgiu uma maneira de fazer uma hierarquia em camadas onde as superiores prestam serviços as inferiores.

6- tarefas : conjunto de ações a serem realizadas para o cumprimento de um objetivo em um determinado tempo.

Processos : são programas que estão sendo executados em um espaço virtual de endereçamento exclusivo.

Threads : é a linha de execução dentro de um processo.

7- No exemplo pratico podemos ver na memória a tarefa dela é armazenar o processo da memória tem obrigatoriamente de passar por três fases: a memorização, o armazenamento e a rememoração e a threads é a execução disso tudo.

8-consiste em uma memória , uma unidade aritmética lógica (ULA) e uma unidade de controle(CU).Ela possibilitou as maquinas armazenarem seus programas no mesmo espaço que os dados assim podendo manipular os programas.

9- Run que está sendo executado no processador, ready ou executável dispõe de todos os recursos que precisa e está pronto para ser executado,sleep ou dormente bloqueado à espera de algum recurso, e só pode ser desbloqueado se receber um sinal de outro processo, zumbi onde um processo é criado por um programa, que por sua vez é finalizado antes de receber o resultado do processo parado Recebeu ordem do administrador para interromper a execução. Será reiniciado se receber um sinal de continuação.

10 – programação sequencial é uma programação que realiza um conjunto predeterminado de comandos de forma sequencial na ordem em que foram feitos.

L1 – ação 1

L2 – ação 2

L3 – ação 3

Assim por diante.

Multiprogramação: uma técnica para maximizar o uso da CPU.

Um exemplo dela é a utilização mais inteligente nos recursos de hardware.

11- Processador é o conjunto da unidade lógica e aritmética, registradores e da unidade de controle. Sua função é executar os programas armazenados na memória principal, buscando suas instruções, examinando-as, e então executando uma após a outra.

O processador é responsável pela realização de uma série de funções:

Busca de instruções e dados na memória;

Programa a transferência de dados entre a memória e os dispositivos de entrada/saída;

Decodifica as instruções;

Realiza as operações lógica e aritméticas;

Responde a sinais enviados por dispositivos de entrada/saída como RESET ou interrupções.

12- Exclusão mútua é um recurso ou técnica que evita que dois processos tenham acesso simultâneo a um recurso compartilhado no processador, isso ocorre, por exemplo, em um semáforo binário, ou seja, que só pode assumir dois valores, 0 e 1. Esse bloqueio acontece sempre antes do recurso ser utilizado, assim só uma thread usará o recurso, e após o recurso ser utilizado o semáforo entra em desuso. As soluções para a exclusão mútua são a solução de Dekker, em que é usada a variável "vez" para realizar o "tie-break", e a solução de Peterson.

13- É o mecanismo programático pelo qual um programa de computador solicita um serviço do núcleo do sistema operacional sobre o qual ele está sendo executado.

14- Uma trap é uma exceção em um processo do usuário. É causado por divisão por zero ou acesso inválido à memória. É também a maneira usual de invocar uma rotina do kernel (uma chamada do sistema) porque é executada com uma prioridade mais alta que o código do usuário. Uma interrupção é algo gerado pelo hardware (dispositivos como disco rígido, placa gráfica, portas de E / S, etc.). Elas são assíncronas (ou seja, não ocorrem em locais previsíveis no código do usuário) ou "passivas", pois o manipulador de interrupções precisa esperar que elas aconteçam eventualmente.

15- É muito importante sobre a questão do desempenho. Chamadas ao sistema requerem desvio de fluxo e tratamento das chamadas que trazem consigo armazenamento e resgate de contexto, coisas que tomam tempo.

16-No exemplo do link onde o Win32 não tem suporte para ele, talvez por falta de informações não enviadas na chamada de sistema ou por falta de parâmetros que não estão no alcance do Win32.

17-A chamada read armazena os bytes e os carrega no buffer, gera o descritor de arquivo, chama a rotina da biblioteca(call), executa a instrução TRAP(instrução que permite acesso ao modo kernel), passa a instrução para um endereço específico do kernel e ativa o endereço específico para os registradores.

LAB 1

1-

```
#include <iostream>

using namespace std;


int main() {

	int n=100000;

	while(n<1000000) {
		cout << n << "\n";

	}



}
```

2-

```
12 if (pid < 0) { /* error occurred */
13 fprintf(stderr, "Fork Failed\n");
14 return 1;
15 }
16 else if (pid == 0) { /* child process */
17 printf("I am the child %d\n",pid);
18 execlp("/bin/ls","ls",NULL);
19 }
20 else { /* parent process */
21 /* parent will wait for the child to complete */
22 printf("I am the parent %d\n",pid);
23 wait(NULL);
```

```
24
25 printf("Child Complete\n");
26 }
```

Serão criado 3 processos