Sistemas Operacionais I

Aula 01

Ciência da Computação

Universidade Federal do Mato Grosso

Prof. Sandino Jardim

O que acontece quando um programa é executado?

- Processador busca, decodifica e executa instruções
- Modelo de Von Neumann: execução sequencial
- Complexidades são escondidas pelo SO

O papel do Sistema Operacional

- Virtualiza recursos físicos
- Gerencia hardware de forma eficiente
- Fornece APIs (system calls)
- Atua como máquina virtual e gerenciador de recursos

Código: cpu.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <assert.h>
#include "common.h"

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "usage: cpu <string>\n");
        exit(1);
    }
    char *str = argv[1];
    while (1) {
        Spin(1);
        printf("%s\n", str);
    }
    return 0;
}
```

Execução de `cpu.c`

```
$ gcc -o cpu cpu.c -Wall
$ ./cpu "A"
A
A
A
^C
```

- Loop infinito com impressão

Virtualização da CPU

```
$ ./cpu A & ./cpu B & ./cpu C & ./cpu D &
```

- Saída:

```
[1] 7353
[2] 7354
[3] 7355
[4] 7356
A
B
D
C
A
B
...
```

Virtualização e abstração

O SO utiliza **virtualização** para transformar recursos físicos (CPU, memória, disco) em formas mais úteis:

- Cada processo “acredita” ter o controle total da CPU e da memória.
- Programas compartilham o mesmo hardware de forma segura e isolada.
- O SO fornece chamadas de sistema (system calls) para que programas interajam com o hardware.

Código `mem.c`

```
int *p = malloc(sizeof(int));
*p = 0;
while (1) {
    Spin(1);
    *p = *p + 1;
    printf("(%d) p: %d\n", getpid(), *p);
}
```

Saída `mem.c`

```
prompt> ./mem & ./mem &
[1] 24113
[2] 24114
(24113) address pointed to by p: 0x200000
(24114) address pointed to by p: 0x200000
(24113) p: 1
(24114) p: 1
(24114) p: 2
(24113) p: 2
(24113) p: 3
(24114) p: 3
(24113) p: 4
(24114) p: 4
...
```

Saída `mem.c`

- Cada instância do programa aloca a mesma posição de memória virtual (ex: 0x200000),
- Na prática cada processo possui seu **próprio espaço de endereçamento virtual**.

Virtualização da Memória

- A memória física é um recurso compartilhado, mas o SO oferece a **abstração de memória exclusiva**. Cada processo tem seu próprio "mundo privado" de memória.
- Isso previne que processos interfiram uns nos outros e melhora a segurança e estabilidade do sistema.

Concorrência

- No exemplo abaixo, dois threads atualizam uma variável `counter` compartilhada:

```
void *worker(void *arg) {
    for (int i = 0; i < loops; i++) counter++;
}

Pthread_create(&p1, NULL, worker, NULL);
Pthread_create(&p2, NULL, worker, NULL);
Pthread_join(p1, NULL);
Pthread_join(p2, NULL);
```

Problemas de Concorrência

```
$ ./threads 100000
Final value: 143012
```

- Isso acontece porque múltiplos threads acessam e modificam `counter` ao mesmo tempo, sem coordenação.
- É necessário usar **mecanismos de sincronização** para garantir a **correção** de programas concorrentes.

Persistência e I/O com arquivos

```
int fd = open("/tmp/file", O_WRONLY|O_CREAT|O_TRUNC, S_IRWXU);
write(fd, "hello world\n", 13);
close(fd);
```

- Essas chamadas de sistema (system calls) permitem ao programa criar, escrever e fechar arquivos.
- O SO garante que os dados sejam armazenados de forma segura e duradoura.

Temas centrais da disciplina

1. **Virtualização**: criar abstrações poderosas de recursos físicos
2. **Concorrência**: lidar com múltiplos processos/threads de forma correta e eficiente
3. **Persistência**: armazenar dados de forma durável mesmo após desligamento

Objetivos de Projeto de um SO

- **Abstração**: facilitar o uso de recursos complexos
- **Eficiência**: evitar desperdício de tempo e espaço
- **Proteção**: isolar processos e evitar acessos indevidos
- **Confiabilidade**: o SO deve ser robusto e resiliente a falhas

História resumida dos Sistemas Operacionais

- Início como bibliotecas de funções
- Evolução para **multiprogramação**
- Desenvolvimento de **UNIX**, multitarefa, proteção de memória
- Adoção em larga escala com PCs e depois smartphones
- Hoje: **Linux**, **Windows**, **macOS**, **Android** — todos com raízes nos mesmos conceitos

Conclusão

O sistema operacional:

- Faz o computador parecer simples de usar
- Gerencia recursos limitados de forma eficiente
- Permite que muitos programas compartilhem o mesmo hardware com segurança

Estudar SO é entender **como os computadores realmente funcionam por dentro**.