Aula 03 – Parte 1
Sistemas Operacionais I

Conceitos Básicos

Prof. Julio Cezar Estrella
jcezar@icmc.usp.br

Material adaptado de

Sarita Mazzini Bruschi

baseados no livro Sistemas Operacionais Modernos de A. Tanenbaum

Roteiro

- Conceitos Básicos
- Chamadas ao Sistema
- Estrutura de Sistemas Operacionais

Conceitos Básicos de Sistemas Operacionais

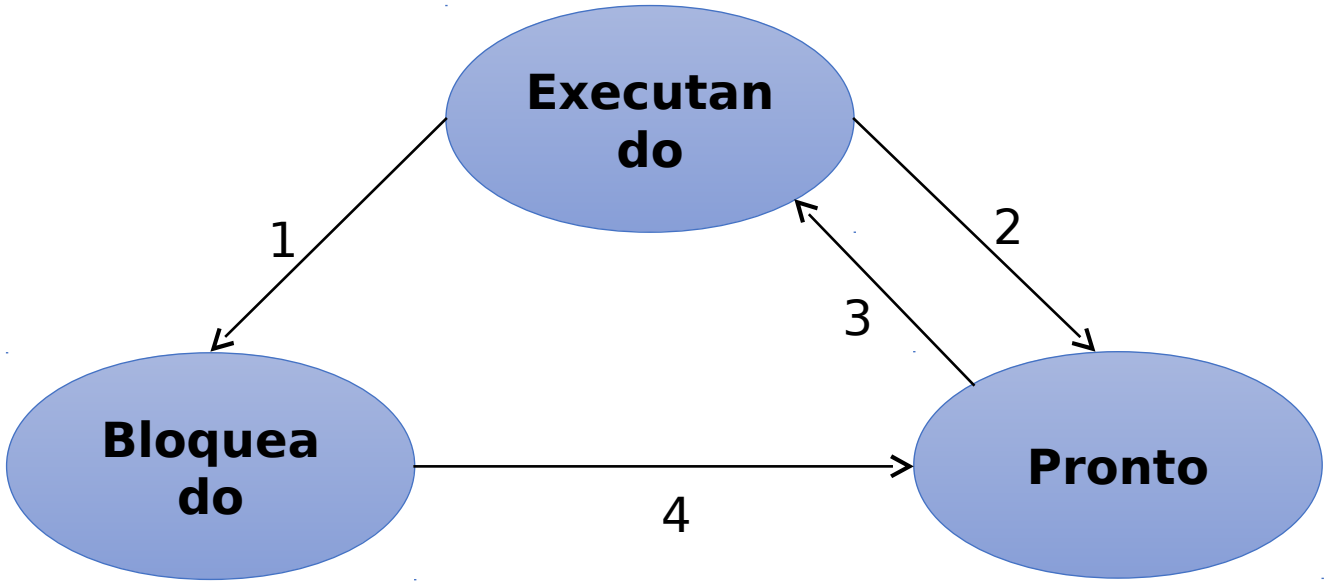
- Principais conceitos:
  - Processo;
  - Memória;
  - Chamadas ao Sistema;

Processos

- Processo: chave do SO;
  - Caracterizado por programas em execução;
  - Cada processo possui:
    - Um espaço de endereço;
    - Uma lista de alocação de memória (mínimo, máximo);
    - Um conjunto de registradores (contador de programa);
  - O Sistema Operacional controla todos os processos;

Processos

- Estados básicos de um processo:



Processos

- Ex.: processo bloqueado (suspenso)
  - Quando o SO suspende um processo *P1* temporariamente para executar um processo *P2*, o processo *P1* deve ser reiniciado exatamente no mesmo estado no qual estava ao ser suspenso. Para tanto, todas as informações a respeito do processo *P1* são armazenadas em uma **tabela de processos** (*process table*). Essa tabela é um vetor ou uma lista encadeada de estruturas.

Processos

- Um processo pode resultar na execução de outros processos, chamados de processos-filhos:
  - Características para a hierarquia de processos:
    - Comunicação (Interação) e Sincronização;
    - Segurança e proteção;
- Escalonadores de processos: processo que escolhe qual será o próximo processo a ser executado;
  - Diversas técnicas para escalonamento de processos;

Processos

- Comunicação e sincronismo entre processos – solução:
  - Semáforos;
  - Monitores;
  - Instruções especiais em hardware;
  - Troca de mensagens;

Gerenciamento de Memória

- Gerenciamento elementar (década de 60)
  - Sistema monoprogramado;
  - Sem paginação:
    - Apenas um processo na memória;
    - Acesso a toda a memória;
- Gerenciamento mais avançado (atualidade)
  - Sistema multiprogramado;
  - Mais de um processo na memória;
  - Chaveamento de processos: troca de processos devido a entrada/saída ou por limite de tempo (sistema de tempo compartilhado);

Gerenciamento de Memória

- Partições Fixas
  - Cada processo é alocado em uma dada partição da memória (pré-definida);
  - Partições são liberadas quando o processo termina;
- Partições Variáveis
  - Memória é alocada de acordo com o tamanho e número de processos;
  - Otimiza o uso da memória;

Roteiro

- Conceitos Básicos
- Chamadas ao Sistema
- Estrutura de Sistemas Operacionais

System Calls – Chamadas ao Sistema

- Interface entre o Sistema Operacional e os programas do usuário;
- As chamadas se diferem de SO para SO, no entanto, os conceitos relacionados às chamadas são similares independentemente do SO;
- Apenas uma chamada de sistema pode ser realizada em um instante de tempo (ciclo de relógio) pela CPU;

Interfaces de um Sistema Operacional

- Interação Usuário ↔ SO:
  - **Shell ou Interpretador de comandos**

- Interação Programas ↔ SO:
  - **Chamadas ao Sistema**

Conceitos Básicos
Chamadas ao Sistema

- **Modos de Acesso**:
  - Modo usuário;
  - Modo *kernel* ou Supervisor ou Núcleo;
- São determinados por um conjunto de bits localizados no registrador de status do processador: PSW (*Program Status Word*);
  - Por meio desse registrador, o hardware verifica se a instrução pode ou não ser executada pela aplicação;
- Protege o próprio *kernel* do Sistema Operacional na RAM contra acessos indevidos;

Conceitos Básicos
Chamadas ao Sistema

- Modo usuário:
  - Aplicações não têm acesso direto aos recursos da máquina, ou seja, ao hardware;
  - Quando o processador trabalha no modo usuário, a aplicação só pode executar **instruções sem privilégios**, **com um acesso reduzido de instruções**;
  - Por que? Para garantir a **segurança** e a **integridade do sistema**;

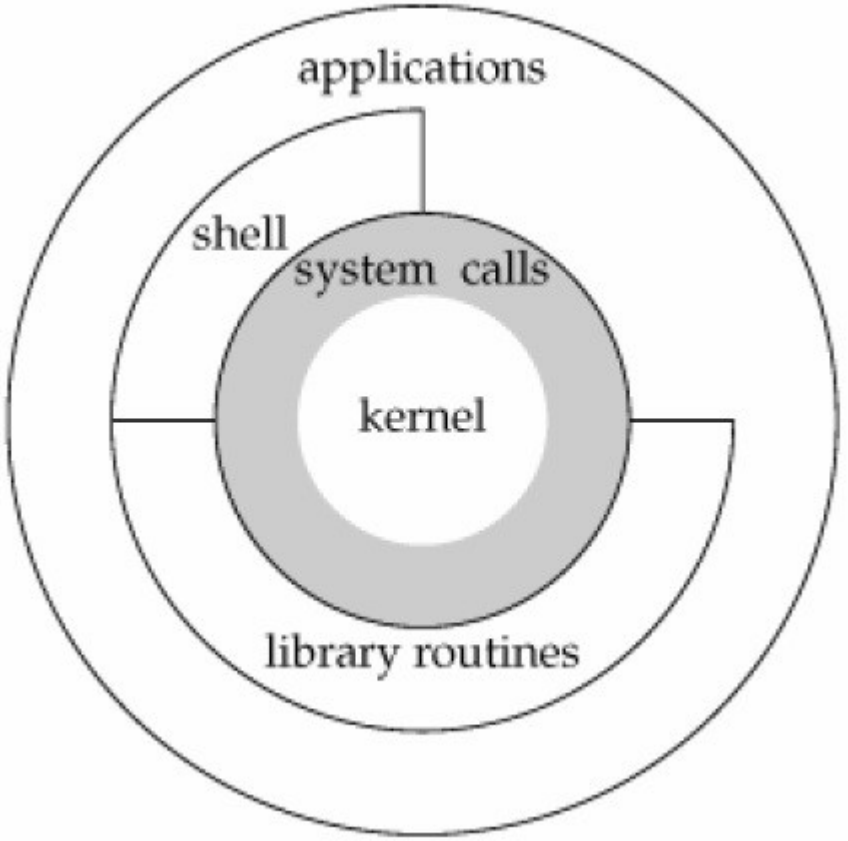
Conceitos Básicos
Chamadas ao Sistema

- Modo *Kernel:*
  - Aplicações têm acesso direto aos recursos da máquina, ou seja, ao hardware;
  - **Operações com privilégios**;
  - Quando o processador trabalha no modo *kernel*, a aplicação tem **acesso ao conjunto total de instruções**;
  - Apenas o SO tem acesso às instruções privilegiadas;

Conceitos Básicos
Chamadas ao Sistema

- Se uma aplicação precisa realizar alguma instrução privilegiada, ela realiza uma **chamada ao sistema *(system call)***, que altera do modo usuário para o modo *kernel*;
- Chamadas de sistemas são a **porta de entrada** para o modo *Kernel*;
  - São a interface entre os programas do usuário no modo usuário e o Sistema Operacional no modo kernel;
  - As chamadas diferem de SO para SO, no entanto, os conceitos relacionados às chamadas são similares independentemente do SO;

Conceitos Básicos
Chamadas ao Sistema

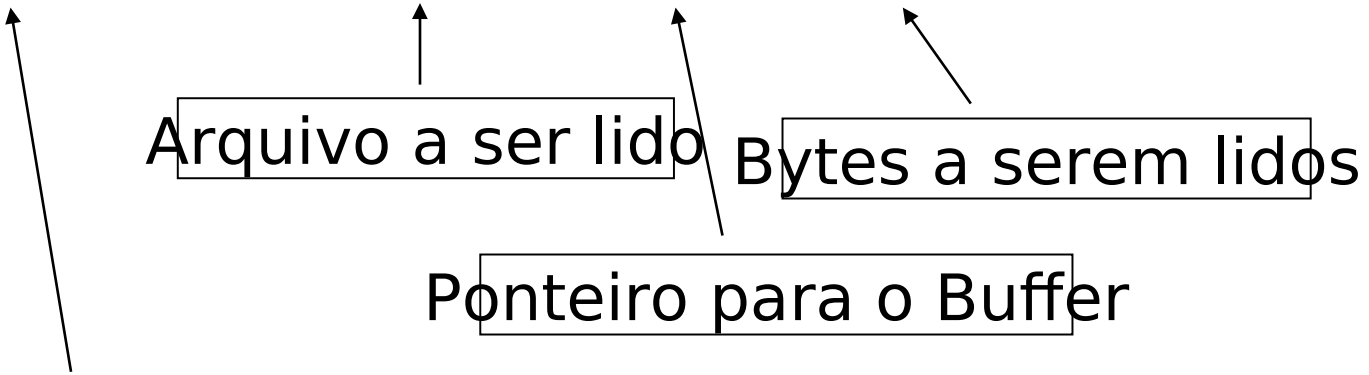


- Fonte: "*Advanced Linux Programming*" Mark Mitchell, Jeffrey Oldham, e Alex Samuel (http://www.advancedlinuxprogramming.com/)

Conceitos Básicos
Chamadas ao Sistema

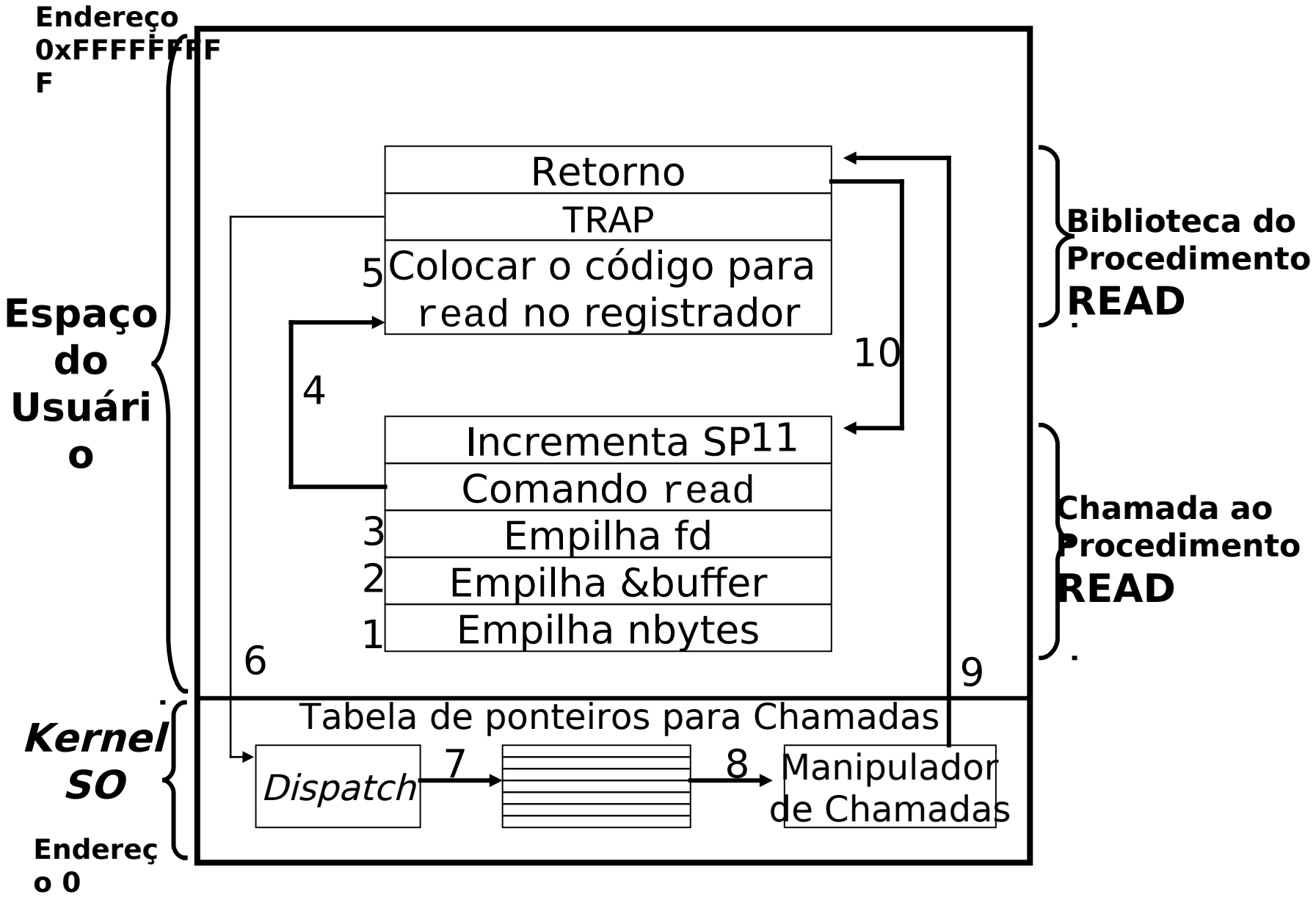
- **TRAP**: instrução que permite o acesso ao modo *kernel*;
- Exemplo:
  - Instrução do UNIX:

```
count = read(fd,buffer,nbytes);
```



O programa sempre deve checar o retorno da chamada de sistema para saber se algum erro ocorreu!!!

Chamadas ao Sistema

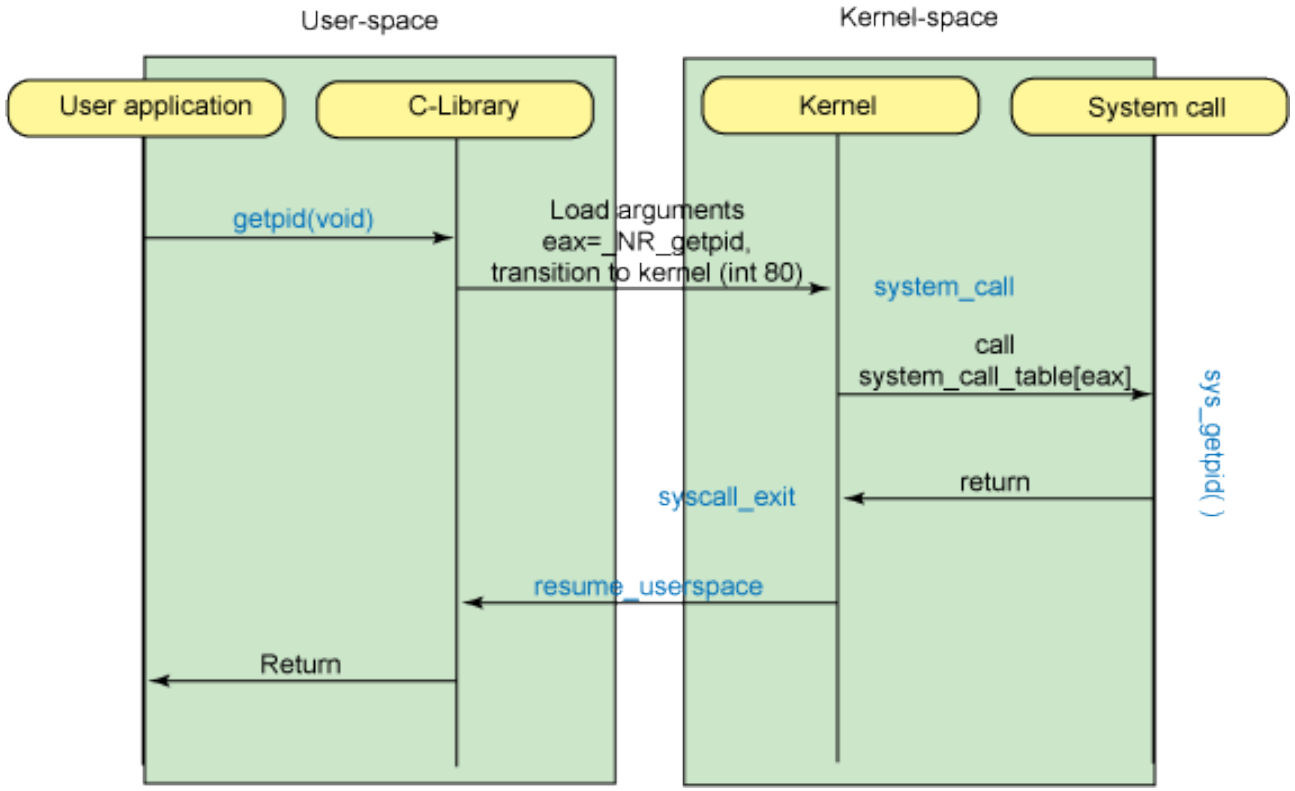


Chamadas ao Sistema

- No Linux, para a arquitetura x86
  - Processo chama uma função da biblioteca libc READ
  - Função da biblioteca coloca o código da chamada do sistema READ no registrador e envia uma interrupção 0x80 para a CPU
  - Nesse momento o código deixa de ser executado em espaço de usuário sendo executado agora no kernel e passa para a rotina de tratamento dessa interrupção que está em /arch/x86/kernel/entry_32.S
  - Essa rotina checa o código que está no registrador e executa a chamada do sistema

Chamadas ao Sistema

- Exemplo da função getpid



Chamada do Sistema getpid [Fonte: IBM]

Chamadas ao Sistema

- No Pentium 4 e sucessores
  - Não utiliza interrupções
  - Utiliza primitivas SYSCALL/SYSRET (AMD), SYSENTER/SYSEXIT (Intel) para mudar para o modo protegido
  - São instruções otimizadas para realizar chamadas de sistema
  - Também armazena o código da chamada do sistema no registrador

Conceitos Básicos
Chamadas ao Sistema

- Exemplos de chamadas da interface:
  - Chamadas para gerenciamento de processos:
    - `Fork (CreateProcess – WIN32)` – cria um processo;
    - Outros exemplos no Posix (***P**ortable **O**perating **S**ystem **I**nterface*)

Gerenciamento de processos

Chamada	Descrição
pid = fork( )	Crie um processo filho idêntico ao processo pai
pid = waitpid(pid, &statloc, options)	Aguarde um processo filho terminar
s = execve(name, argv, environp)	Substitua o espaço de endereçamento do processo
exit(status)	Termine a execução do processo e retorne o estado

Exemplo – fork()

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

main()
{
        int pid;
        pid=fork();
        if ( pid < 0 ) { fprintf(stderr,"erro\n"); exit(1); }
        if ( 0 == pid )
                printf("FILHO: \t id is %d, pid (valor)is %d\n",getpid(), pid);
        else
                printf("PAI: \t id is %d, pid (filho)is %d\n", getpid(), pid);
        /* este comando executado duas vezes..*/
        system("date");
}
```

Exemplo – execve()

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
        char *args[ ] = {"/bin/ls", NULL};
        if (execve("/bin/ls", args, NULL) == -1) {
                perror("execve");
                exit(EXIT_FAILURE);
        }

        puts("não deveria ter chegado até aqui");
        exit(EXIT_SUCCESS);
}
```

Exemplo – fork() + execve()

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>

int main (int argc, char *argv[], char *envp[])
{
  int retval ;

  printf ("Ola, sou o processo %5d\n", getpid()) ;

  retval = fork () ;

  printf ("[retval: %5d] sou %5d, filho de %5d\n",
        retval, getpid(), getppid()) ;

if ( retval < 0 )
  {
    perror ("Erro: ") ;
    exit (1) ;
  }
  else
    if ( retval > 0 )
      wait (0) ;
    else
    {
      execve ("/bin/date", argv, envp) ;
      perror ("Erro") ;
    }

  printf ("Tchau de %5d!\n", getpid()) ;

  exit (0) ;
}
```

Conceitos Básicos
Chamadas ao Sistema

- Exemplos de chamadas da interface :
  - Chamadas para gerenciamento de diretórios:
    - `Mount` – monta um diretório;
  - Chamadas para gerenciamento de arquivos:
    - `Close (CloseHandle – WIN32)` – fechar um arquivo;
  - Outros exemplos no Posix

Gerenciamento do sistema de diretório e arquivo

Chamada	Descrição
s = mkdir(name, mode)	Crie um novo diretório
s = rmdir(name)	Remova um diretório vazio
s = link(name1, name2)	Crie uma nova entrada, name2, apontando para name1
s = unlink(name)	Remova uma entrada de diretório
s = mount(special, name, flag)	Monte um sistema de arquivo
s = umount(special)	Desmonte um sistema de arquivo

Conceitos Básicos
Chamadas ao Sistema

- Exemplos de chamadas da interface :
  - Outros tipos de chamadas:
    - `Chmod`: modifica permissões;
  - Outros exemplos no Posix

Diversas

Chamada	Descrição
s = chdir(dirname)	Altere o diretório de trabalho
s = chmod(name, mode)	Altere os bits de proteção do arquivo
s = kill(pid, signal)	Envie um sinal a um processo
seconds = time(&seconds)	Obtenha o tempo decorrido desde 1º de janeiro de 1970

Conceitos Básicos
Chamadas ao Sistema

- Chamadas da interface: Unix x Windows:
  - **Unix**
    - Chamadas da interface muito semelhantes às chamadas ao sistema
    - Aproximadamente 100 chamadas a procedimentos
  - **Windows**
    - Chamadas da interface totalmente diferente das chamadas ao sistema
    - APIWin32 (Application Program Interface)
      - Padrão de acesso ao sistema operacional
      - Facilita a compatibilidade
      - Possui milhares de procedimentos

Conceitos Básicos
Chamadas ao Sistema

- Exemplos de chamadas da interface: Unix e API Win32

Unix	Win32	Descrição
fork	CreateProcess	Crie um novo processo
waitpid	WaitForSingleObject	Pode esperar um processo sair
execve	(none)	CrieProcesso = fork + execve
exit	ExitProcess	Termine a execução
open	CreateFile	Crie um arquivo ou abra um arquivo existente
close	CloseHandle	Feche um arquivo
read	ReadFile	Leia dados de um arquivo
write	WriteFile	Escreva dados para um arquivo
lseek	SetFilePointer	Mova o ponteiro de posição do arquivo
stat	GetFileAttributesEx	Obtenha os atributos do arquivo
mkdir	CreateDirectory	Crie um novo diretório
rmdir	RemoveDirectory	Remova um diretório vazio
link	(none)	Win32 não suporta ligações (link)
unlink	DeleteFile	Destrua um arquivo existente
mount	(none)	Win32 não suporta mount
umount	(none)	Win32 não suporta mount
chdir	SetCurrentDirectory	Altere o diretório de trabalho atual
chmod	(none)	Win32 não suporta segurança (embora NT suporte)
kill	(none)	Win32 não suporta sinais
time	GetLocalTime	Obtenha o horário atual