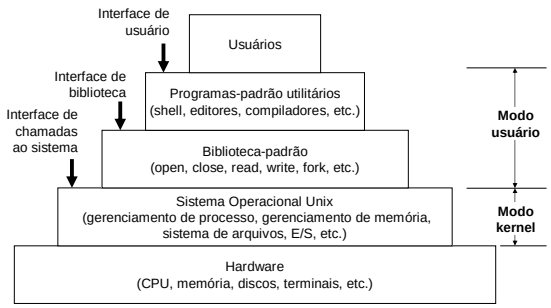
Estudo de Caso: Sistema Operacional Unix

Prof. Alexandro M. S. Adário
URI – Campus de Erechim

Breve Histórico do Unix e do Linux

- Versão original desenvolvida por Ken Thompson (Bell)
  - Assembly para PDP-7, logo portada para o PDP-11
  - Baixa Portabilidade → Mudança para código em "B"
  - Denis Ritchie desenvolve o C, escreve o compilador
- Unix é licenciado para universidades pela AT&T
  - Berkeley desenvolve o 1BSD que evolui para produtos de grande sucesso (4BSD e releases)
  - Microsoft lança o Xenix
  - AT&T consegue entrar no mercado e evolui até o System V
- POSIX (Portable Operating System) é uma iniciativa de padronização
- Minix: versão didática desenvolvida por Tanenbaum
- Linux é baseado no Minix, como um clone do Unix.

Camadas de um Sistema Unix



Estrutura do Núcleo do 4.4BSD

<table>
<tr><td colspan="5">Chamadas ao sistema</td><td colspan="3">Interrupção e traps</td></tr>
<tr><td colspan="2">Tratamento de terminais</td><td>Sockets</td><td>Nomeação de arquivos</td><td>Mapea-mento</td><td>Faltas de página</td><td>Tratamento de sinais</td><td>Criação e término de processos</td></tr>
<tr><td rowspan="2">Terminais tty naturais</td><td>Terminais tty preparados</td><td>Protocolos de rede</td><td>Sistemas de arquivos</td><td colspan="2">Memória Virtual</td><td colspan="2" rowspan="2">Escalonamento de processos</td></tr>
<tr><td>Tratamento de linhas</td><td>Roteamento</td><td>Cache de buffer</td><td colspan="2">Cache de páginas</td></tr>
<tr><td colspan="2">Dispositivos de caracteres</td><td>Drivers de Rede</td><td colspan="2">Drivers de disco</td><td colspan="2">Despacho de processos</td></tr>
<tr><td colspan="8">Hardware</td></tr>
</table>

Sinais Exigidos pela Norma POSIX

Sinal	Efeito
SIGABRT	Enviado para abortar um processo e forçar descarregamento de memória (*core dump*)
SIGALRM	O relógio do alarme disparou
SIGFPE	Ocorreu um erro de ponto flutuante (por exemplo, divisão por 0)
SIGHUP	A linha telefônica que o processo estava usando caiu
SIGILL	O usuário pressionou a tecla DEL para interromper o processo
SIGQUIT	O usuáio pressionou uma tecla solicitando descarregamento de memória
SIGKILL	Enviado para matar um processo (não pode ser capturado ou ignorado)
SIGPIPE	O processo escreveu em um pipe que não tem leitores
SIGSEGV	O processo referenciou um endereço de memória inválido
SIGTERM	Usado para requisitar que um processo termine elegantemente
SIGUSR1	Disponível para propósitos definidos pela aplicação
SIGUSR2	Disponível para propósitos definidos pela aplicação

Chamadas de Sistema – Gerenc. Processos

Chamada ao sistema	Descrição
pid = fork( )	Cria um processo filho idêntico ao pai
pid = waitpid(pid, &statloc, opts)	Espera o processo filho terminar
s = execve(name, argv, envp)	Substitui a imagem da memória de um processo
exit(status)	Termina a execução de um processo e retorna o status
s = sigaction(sig, &act, &oldact)	Define a ação a ser tomada nos sinais
s = sigreturn(&context)	Retorna de um sinal
s = sigprocmask(how, &set, &old)	Examina ou modifica a máscara do sinal
s = sigpending(set)	Obtém o conjunto de sinais bloqueados
s = sigsuspend(sigmask)	Substitui a máscara de sinal e suspende o processo
s = kill(pid, sig)	Envia um sinal para um processo
residual = alarm(seconds)	Ajusta o relógio do alarme
s = pause( )	Suspende o chamador até o próximo sinal

Exemplo de Chamadas de Sistema

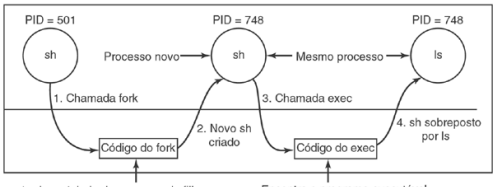
```
while (TRUE) {                          /* repete para sempre
     type_prompt( );                    /* mostra o prompt na tela
     read_command(command, params);     /* lê a linha de entrada do teclado

     pid = fork( );                     /* cria um processo filho
     if (pid < 0) {
          printf("Unable to fork0);     /* condição de erro
          continue;                     /* repete o laço
     }

     if (pid != 0) {
          waitpid (–1, &status, 0);     /* pai espera o filho
     } else {
          execve(command, params, 0);   /* filho traz o trabalho
     }
}
```

Implementa uma shell básica simplificada

Implementação de Threads POSIX

Chamadas a thread	Descrição
pthread_create	Cria um novo thread no espaço de endereço do chamador
pthread_exit	Termina o thread chamador
pthread_join	Espera pelo término de um thread
pthread_mutex_init	Cria um novo mutex
pthread_mutex_destroy	Destrói um mutex
pthread_mutex_lock	Impede um mutex
pthread_mutex_unlock	Libera um mutex
pthread_cond_init	Cria uma variável de condição
pthread_cond_destroy	Destrói uma variável de condição
pthread_cond_wait	Espera por uma variável de condição
pthread_cond_signal	Libera um thread que está à espera sobre uma variável de condição

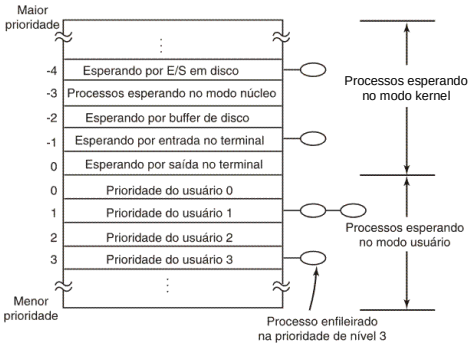
Execução de um "ls" na Shell



Aloca entrada na tabela de processo do filho
Preenche entrada do filho com dados do pai
Aloca pilha e área de usuário para o filho
Preenche a área de usuário do filho com dados do pai
Aloca PID para o filho
Ajusta filho para compartilhar código do pai
Copia tabelas de páginas para dados e pilha
Ajusta compartilhamento de arquivos abertos
Copia registradores do pai para o filho

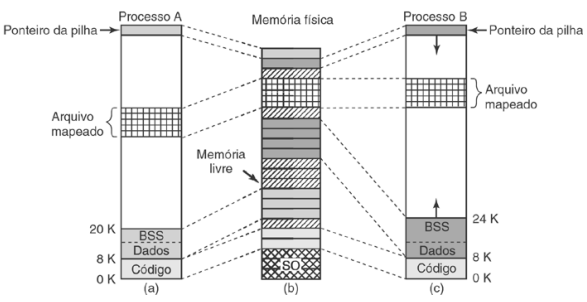
Encontra o programa executável
Verifica a permissão de execução
Lê e verifica o cabeçalho
Copia argumentos e variáveis de ambiente para o núcleo
Libera o antigo espaço de endereçamento
Aloca novo espaço de endereçamento
Copia argumentos e variáveis de ambiente para a pilha
Reinicializa os sinais
Inicializa os registradores

Fila de Escalonamento Unix

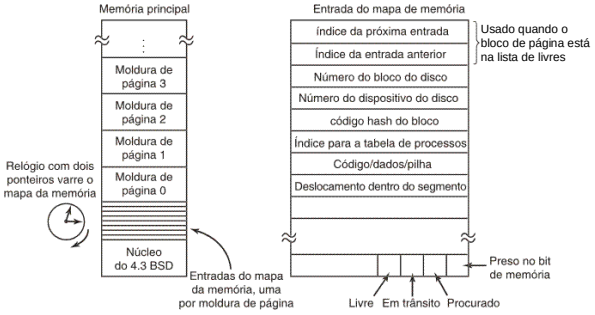


O escalonador do Unix é baseado em uma estrutura de fila multinível

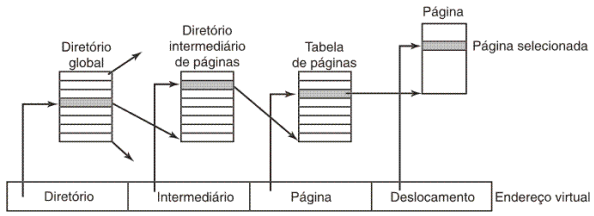
Gerência de Memória e File Sharing



Sistema de Paginação do 4BSD



Sistema de Paginação do Linux



O Linux usa tabelas de páginas de três níveis

Entrada e Saída no Unix