


© 1997-2009 - Volnys Bernal 1

Processos

Volnys Borges Bernal
volnys@lsi.usp.br
<http://www.lsi.usp.br/~volnys>

Laboratório de Sistemas Integráveis
<http://www.lsi.usp.br/>




© 1997-2009 - Volnys Bernal 2

Agenda

- ❑ Programa x Processo
 - ❖ Programa
 - ❖ Processo
- ❑ Concorrência x Paralelismo
- ❑ Áreas de memória de um processo
- ❑ Exemplo: processos no ambiente UNIX
- ❑ Troca de contexto
- ❑ Ciclo de vida dos processos
- ❑ Processos no ambiente UNIX
- ❑ Hierarquia de ativação de processos

© 1997-2009 - Volnys Bernal 3

Programa x Processo




© 1997-2009 - Volnys Bernal 4

Programa x Processo

- ❑ Programa
 - ❖ Sequência de comandos e dados definidos para realizar uma tarefa
- ❑ Processo
 - ❖ Execução do programa

© 1997-2009 - Volnys Bernal 5

Programa



© 1997-2009 - Volnys Bernal 6

Programa

- ❑ Sequência de instruções e dados que podem ser executados para realizar uma determinada tarefa
- ❑ Programa Fonte
 - ❖ Codificação que contém comandos e dados
 - ❖ Sintaxe da codificação é realizada em uma linguagem de baixo nível (assembler) ou alto nível (C, Pascal, Fortran, Java, ...)
- ❑ Programa Executável
 - ❖ Codificação que contém as instruções de máquina e dados
 - ❖ Sintaxe da codificação é a linguagem de máquina (linguagem do processador).

© 1997-2009 - Volnys Bernal 7

Programa

Programa Fonte

Compilador

Programa executável

Armazenado em arquivo

Contém uma de dados e comandos em linguagem de alto nível representados como uma seqüência de caracteres ASCII.

Armazenado em arquivo

Contém uma seqüência de instruções e dados codificados em linguagem de máquina

© 1997-2009 - Volnys Bernal 8

Exercício

(1) Quais são as principais informações que um programa fonte contém?

(2) Quais são as principais informações que um programa executável contém?

© 1997-2009 - Volnys Bernal 9

Programa

□ Seções de um arquivo de programa executável

- ❖ Área de código
 - Contém as instruções em linguagem de máquina
- ❖ Área de dados
 - Dados iniciados
 - Contém as variáveis e estruturas globais que possuem valores definidos inicialmente
 - Constantes
 - Dados não iniciados
 - Variáveis e estruturas não inicializadas
- ❖ Tabela de símbolos
 - Tabela que contém o símbolo (nome de função, nome de variável, nome de estrutura, nome de *label*, nome de constante, etc) e o endereço de memória virtual de cada símbolo.

© 1997-2009 - Volnys Bernal 10

Programa

□ Arquivo executável

- ❖ Denominação geralmente atribuída ao arquivo que armazena um programa executável
- ❖ O formato do arquivo executável varia de acordo com o sistema operacional. De forma geral possui a seguinte estrutura:

▪ Tabela que contém o símbolo (nome de função, nome de variável, nome de estrutura, nome de *label*, nome de constante, etc) e o endereço de memória virtual de cada símbolo.

Cabeçalho
Área de Código
Área de Dados iniciados e constantes
Tabela de Símbolos

Estrutura geral de um arquivo executável

© 1997-2009 - Volnys Bernal 11

Programa

□ Arquivo executável

- ❖ Cabeçalho
 - Estrutura localizada no início do arquivo
 - Informa:
 - Para a área de código
 - Tamanho da área de código
 - Início da área de código no arquivo
 - Endereço inicial de memória virtual na qual esta área deve ser carregada
 - Para a área de dados iniciados
 - Tamanho da área de dados iniciados
 - Início da área de dados iniciados no arquivo
 - Endereço inicial de memória virtual a partir da qual a área de dados iniciados deve ser carregada
 - Para a área de dados não iniciados
 - Tamanho da área de dados não iniciados
 - Endereço inicial de memória virtual a partir da qual a área de dados não iniciados deve ser alocada.
 - Para a área da pilha de execução:
 - Endereço inicial de memória virtual a partir da qual a pilha de execução será alocada
 - EntryPoint
 - Endereço da primeira instrução a ser executada.

© 1997-2009 - Volnys Bernal 12

Programa

□ Arquivo executável

- ❖ Área de código
 - Contém a seqüência de instruções em linguagem de máquina a serem executadas
- ❖ Área de dados iniciados
 - Contém
 - Variáveis e estruturas globais que possuem valores definidos inicialmente
 - Constantes
- ❖ Tabela de símbolos
 - Tabela que contém o símbolo (nome de função, nome de variável, nome de estrutura, nome de *label*, nome de constante, etc) e o endereço de memória virtual de cada símbolo.

Exercício

- (3) Compile e execute o programa fatorial:
- ❖ mkdir <dir>
 - ❖ cd <dir>
 - ❖ cp -volnys/public/so/fatorial.c .
 - ❖ ls -l
 - ❖ cat fatorial.c
 - ❖ cc -o fatorial fatorial.c
 - ❖ ./fatorial
- (4) Em relação ao programa fatorial.c, identifique as variáveis ou estruturas relacionadas a:
- ❖ Dado global inicializado
 - Variável
 - Constante
 - ❖ Dado global não inicializado
 - ❖ Dado local
- (5) Utilize o programa nm (nm -nA fatorial) e relacione as informações relativas às
- ❖ Áreas do programa/processo
 - ❖ Entry point (instrução inicial a ser executada)
 - ❖ Símbolos e seus endereços de memória virtual

Exercício

```
#include <stdio.h>
char versao[] = "2.1";
int n;
int resultado;

int fatorial (int x)
{
    int y;
    if (x <= 1)
        return(1);
    else
    {
        y = x * fatorial(x-1);
        return(y);
    }
}

int main(int argc, char **argv)
{
    printf("Programa fatorial, versao %s \n", versao);
    printf("Entre com o valor: ");
    scanf("%d",&n);
    resultado = fatorial(n);
    printf("Resultado: %d \n",resultado);
}
```

Exercício

❑ Compilação

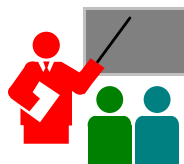
- ❖ Para compilar: `cc -o fatorial fatorial.c`
- Compilador C* *output* *Nome do fonte*
- ❖ Para executar: `./fatorial`

Programa

❑ Exemplo de programas no UNIX

- ❖ O diretório /bin contém vários utilitários do sistema.
- ❖ Listando este diretório (`ls -l /bin`) é possível identificar os arquivos executáveis de alguns programas utilitários:
 - utilitário ls é armazenado no arquivo /bin/ls
 - utilitário cat é armazenado no arquivo /bin/cat
 - utilitário csh é armazenado no arquivo /bin/tcsh

Processo



Processo

- ❑ Processo
- ❖ Um programa sendo executado
 - ❖ Possui um contexto (informações) como:
 - Informações de controle
 - Geralmente armazenadas na tabela de processos
 - Informações:
 - identificação única (pid - process identification)
 - Registradores
 - Estado do processo
 - Identificação do usuário
 - Terminal do qual foi disparado
 - Áreas de memória
 - Área de código
 - Área de dados
 - Área da pilha de execução
 - Outras áreas de memória

© 1997-2009 - Volnys Bernal 19

Processo

- **Processo**
 - ❖ Um programa sendo executado
 - ❖ Possui um contexto (informações) como:
 - Contexto de software
 - Espaço de endereçamento
 - Área de código
 - Área de dados
 - Área da pilha de execução
 - Informações de controle mantidas pelo S.O.
 - Identificação do processo (pid)
 - Identificação do usuário dono do processo
 - Estado do processo
 - ...
 - Contexto de hardware (valores dos registradores)
 - PC (program counter - contador de programa)
 - SP (stack pointer - ponteiro para a pilha de execução)
 - ST (status - estado)
 - Registradores de números inteiros e ponto flutuante

© 1997-2009 - Volnys Bernal 20

Processos

- **Exemplo:**
 - ❖ Tabela de processos do UNIX

<ul style="list-style-type: none"> ❖ Identificação do processo (PID) ❖ Valores de registradores <ul style="list-style-type: none"> • PC, SP, Status, ... ❖ Estado do processo ❖ Data/hora de disparo ❖ Tempo de CPU ocupado ❖ Tempo de CPU ocupado pelos filhos ❖ Hora do próximo alarme ❖ Sinais penderetes ❖ Localização das áreas de memória: <ul style="list-style-type: none"> • Área de código (text) • Área de dados • Área de dados dinâmica • Área da pilha de execução 	<ul style="list-style-type: none"> ❖ Estado de retorno ❖ Estado do sinal ❖ Processo pai ❖ Identificação do "process group" ❖ Identificação do usuário (UID) ❖ Identificação do grupo do usuário (GID) ❖ Effective UID ❖ Effective GID ❖ Umask ❖ Diretório raiz ❖ Diretório de trabalho ❖ Descritores de arquivos abertos
--	--

© 1997-2009 - Volnys Bernal 21

Exercício

(6) Uma das áreas criadas alocadas quando um processo é disparado é a pilha de execução. A pilha de execução possibilita realizar o controle de ativação das subrotinas. A cada ativação de uma subrotina é criado um "quadro" (relacionado à esta instância da execução da subrotina) na pilha de execução do processo. Neste quadro ficam localizadas, dentre outras, as seguintes informações :

- ❖ argumentos da subrotina;
- ❖ endereço de retorno da subrotina;
- ❖ variáveis locais da subrotina;

Simule a execução do programa fatorial com o valor 3, mostrando a evolução dos valores das variáveis globais e da pilha da execução.

© 1997-2009 - Volnys Bernal 22

Processos

- **Visão geral dos processos**
 - ❖ Processo é uma abstração criada pelo Sistema Operacional
 - ❖ O sistema operacional é o responsável pelo gerenciamento dos processos no sistema.


© 1997-2009 - Volnys Bernal 23

Processos

- **Visão geral dos procesos**
 - ❖ Sistemas de computação modernos criam a ilusão de que vários processos (aplicações) executam ao mesmo tempo no sistema.
 - ❖ Na realidade, em sistemas monoprocessadores, em um determinado instante existe somente um programa sendo executado pela CPU.
 - ❖ O sistema operacional gerencia o uso da CPU de forma que seja executado "um pouco" de cada processo por vez. Este chaveamento entre os processos é tão rápido que cria a ilusão de que os processos executam simultaneamente.

© 1997-2009 - Volnys Bernal 24

Concorrência x Paralelismo



© 1997-2009 - Volnys Bernal 25

Concorrência x Paralelismo

- ❑ **Concorrência**
 - ❖ Pseudo paralelismo
 - ❖ Ilusão de execução simultânea de processos devido ao rápido chaveamento entre suas execuções em uma única CPU
- ❑ **Paralelismo**
 - ❖ Paralelismo real
 - ❖ Em sistemas com mais que uma CPU cada uma pode estar executando um processo efetivamente em paralelo.

© 1997-2009 - Volnys Bernal 26

Áreas de memória de um processo

© 1997-2009 - Volnys Bernal 27

Áreas de memória de um processo

© 1997-2009 - Volnys Bernal 28

Áreas de memória de um processo: Exemplo UNIX

Seção do arquivo executável	Segmento do sistema operacional	Espaço de endereçamento virtual do processo	
Cabeçalho		0	
Código	Segmento de código	Código	Seqüência de instruções
Dados iniciados		Data	Dados globais iniciados
Tabela de símbolos	Segmento de dados	BSS	Dados globais não iniciados
		Heap	Alocação dinâmica
	Segmento da pilha de execução	Pilha de execução	Controle de execução de subrotinas: variáveis locais, endereços de retorno, parâmetros das funções
		N	

© 1997-2009 - Volnys Bernal 29

Exemplo: Processos no ambiente UNIX

© 1997-2009 - Volnys Bernal 30

Exemplo: Processos no ambiente UNIX

- ❑ **Utilitário ps**
 - ❖ **Descrição**
 - "Process Status"
 - Permite mostrar informações dos processos
 - ❖ **Sintaxe**

```
ps [fuxa]
```
 - ❖ **Opções**
 - x "x", inclui processos que não possuem terminal de controle
 - a "all", mostra todos processo, inclusive de outros usuários
 - f mostra relação pai-filho
 - u "user oriented output": mostra campos USER, %CPU, %MEM, SZ, RSS and START

Exemplo:
Processos no ambiente UNIX

© 1997-2009 - Volnys Bernal 31

❑ Exemplos

```
(terra|jose) ps x
(terra|jose) ps xa
(terra|jose) ps xau
PID TTY STAT TIME COMMAND
  1 ? S 0:03 init
  2 ? SW 0:00 (kflushd)
  3 ? SW< 0:00 (kswapd)
  4 ? SW 0:00 (md_thread)
221 ? S 0:00 crond
232 ? S 0:00 portmap
210 ? S 0:00 /usr/sbin/atd
446 2 S 0:00 /bin/login -- alunol
464 2 S 0:00 -csh
486 2 R 0:00 ps xa
(terra|jose)
```

Exemplo:
Processos no ambiente UNIX


© 1997-2009 - Volnys Bernal 32

❑ Informações apresentadas

USER	usuário dono do processo
PID	<i>process identification</i> - identificação do processo
%CPU	porcentagem de tempo de CPU consumido recentemente
%MEM	porcent. da memória real (páginas) consumida recentemente
SIZE	<i>size</i> - tamanho dos segmentos de dados e pilha (Kbytes)
RSS	<i>resident set size</i> - memória efetivamente alocada (kbytes)
TT	<i>tty</i> - terminal de controle, terminal do qual foi disparado
STAT	<i>state</i> - estado do processo
START	horário de disparo
TIME	tempo (em seg.) consumido pelo processo desde seu início
COMMAND	linha de comando

© 1997-2009 - Volnys Bernal 33

Troca de Contexto



© 1997-2009 - Volnys Bernal 34

Troca de Contexto

❑ Contexto de um processo:

- ❖ É o conjunto de informações relevantes à execução do processo:
- ❖ Contexto de Hardware:
 - Registradores da CPU
- ❖ Contexto de Software:
 - Área de código
 - Área de dados
 - Área da pilha de execução
 - Identificação do processo (pid – process identification)
 - Estado
 - Prioridade
 - Conjunto de arquivos abertos
 -

© 1997-2009 - Volnys Bernal 35

Troca de contexto

❑ Contexto de um processo:

- ❖ É o conjunto de informações relevantes à execução do processo:
 - Contexto de software
 - Espaço de endereçamento
 - Área de código
 - Área de dados
 - Área da pilha de execução
 - Informações de controle mantidas pelo S.O.
 - Identificação do processo (pid)
 - Identificação do usuário dono do processo
 - Identificação do terminal do qual foi disparado
 - Estado do processo
 - ...
 - Contexto de hardware (valores dos registradores da CPU)
 - PC (program counter - contador de programa)
 - SP (stack pointer – ponteiro para a pilha de execução)
 - ST (status – estado)
 - Registradores de números inteiros e ponto flutuante

© 1997-2009 - Volnys Bernal 36


Troca de Contexto

❑ Troca de contexto

- ❖ Atividade de mudança de contexto de processos em um ambiente de computação
- ❖ Atividade realizada pelo Sistema Operacional
- ❖ Envolve a troca de contexto de hardware e software de um processo:
 - Salvamento do contexto do processo corrente
 - Retomada do contexto do outro processo

© 1997-2009 - Volnys Bernal 37

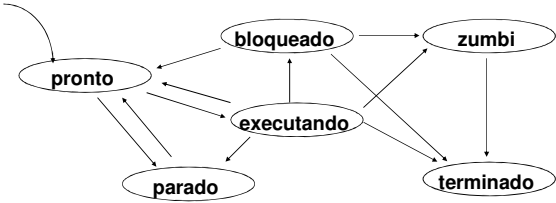
Ciclo de Vida dos Processos



© 1997-2009 - Volnys Bernal 38

Ciclo de Vida dos Processos

- ❑ Os processos, assim como qualquer entidade viva, possui um ciclo de vida:
 - ❖ Nasce, vive e morre
- ❑ Estados de um processo
 - ❖ Diagrama simplificado de transição de estados:




© 1997-2009 - Volnys Bernal 39

Ciclo de Vida dos Processos

- ❑ Estados dos Processos
 - ❖ Pronto
 - O processo está pronto para executar. Não está executando porque a CPU está sendo utilizada por outro processo.
 - ❖ Executando
 - O processo está utilizando a CPU no momento
 - ❖ Bloqueado
 - O processo não pode continuar sua execução enquanto não ocorrer o evento pelo qual espera (ex, leitura de disco, ...)
 - ❖ Parado
 - O processo foi momentaneamente parado pelo usuário ou operador
 - ❖ Zumbi
 - O processo já terminou mas não foram liberadas suas informações de controle
 - ❖ Terminado
 - Processo já terminou e toda informação de controle foi liberada

© 1997-2009 - Volnys Bernal 40

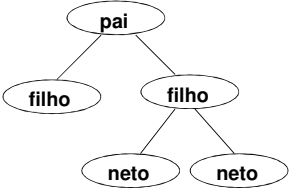
Hierarquia de ativação de processos



© 1997-2009 - Volnys Bernal 41

Hierarquia de ativação de processos

- ❑ Processos pai e filho
 - ❖ Seja um processo A que cria um processo B
 - ❖ O processo A é chamado processo pai do processo B
 - ❖ O processo B é chamado processo filho do processo A



© 1997-2009 - Volnys Bernal 42

Exercício

(7) Utilize o comando “ps xa” do sistema Linux e relacione os processos ativos em sua máquina.

(8) Utilize o comando “ps fxa” e descreva a hierarquia de ativação dos processos ativos em sua máquina.

© 1997-2009 - Volnys Bernal 43

Exercício

(9) Seja o programa “fork1.c” mostrado no slide a seguir. Execute este programa e explique o que o programa realiza.

❖ Para compilar:

```

cc -o fork1 fork1.c
  
```

Compilador *output* *Nome do fonte*
C

❖ Para executar:

```

./fork1
  
```

© 1997-2009 - Volnys Bernal 44

Exercício

```

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

int i = 1;

int main()
{
    pid_t status;

    printf("Programa de disparo de processos filho. \n");
    status = fork();
    printf("Apos a chamada fork(). \n");
    while (1)
    {
        printf(" Status = %5d, i = %d \n",status,i);
        i++;
        sleep(2);
    }
}
  
```

© 1997-2009 - Volnys Bernal 45

Exercício

□ Duplicação de processos

© 1997-2009 - Volnys Bernal 46

Exercício

□ Duplicação de processos

© 1997-2009 - Volnys Bernal 47

Exercício

(10) Seja o programa “fork2.c” mostrado no slide a seguir. Execute este programa e explique o que o programa realiza.

❖ Para compilar:

```

cc -o fork2 fork2.c
  
```

Compilador *output* *Nome do executável*
C

❖ Para executar:

```

./fork2
  
```

© 1997-2009 - Volnys Bernal 48

Exercício


```

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

int main() {
    pid_t pid;
    printf("Programa de disparo de processos filho. \n");
    pid = fork();
    if (pid == 0) {
        printf("Processo filho executando. \n");
        while (1) {
            printf("Filho. \n");
            sleep(2);
        }
    }
    else {
        printf("Processo pai executando: pid do filho = %d\n",pid);
        while (1) {
            printf("Pai. \n");
            sleep(2);
        }
    }
}
  
```


© 1997-2009 - Volnys Bernal 49

Bibliografia



© 1997-2009 - Volnys Bernal 50

Bibliografia

- **Sistemas Operacionais Modernos**
 - ❖ **Andrews Tanenbaum**

- **Bibliografia complementar:**
 - ❖ **The Design of the UNIX Operating System**
 - Maurice J. Bach
 - Prentice-Hall Software Series, 1986