Sistemas Operacionais I

Apresentação do Laboratório de Sistemas Operacionais

Objetivos do Laboratório de SO

- Relacionar a teoria de sistemas operacionais quando aplicada a sistemas operacionais reais
- Operar diferentes tipos de sistemas operacionais, com interfaces e propósitos variados
- Desenvolver programas que explorem os mais variados serviços oferecidos por um sistema operacional, especialmente comunicação e sincronização entre processos

Conteúdo

- Comandos básicos do Shell
- Linguagem C
- Gerência de Arquivos
- Gerência de Processos
- Sinais no Unix
- Comunicação entre Processos e Sincronização: Pipes
- Comunicação entre Processos e Sincronização: FIFOs e Memória Compartilhada
- Comunicação entre Processos e Sincronização: Semáforos e Queues
- Threads (Java e Pthreads)

Laboratório de SO

- Aplicação dos conceitos apresentados nas aulas de teoria de sistemas operacionais
- Uso de recursos do SO (chamadas ao sistema)
- Resolução de problemas de comunicação e programação concorrente
- Exercícios de implementação em todas as aulas
  - Em grupo ou individual
- Uso da linguagem C
- Entrega dos exercícios através do moodle.

Laboratório de SO

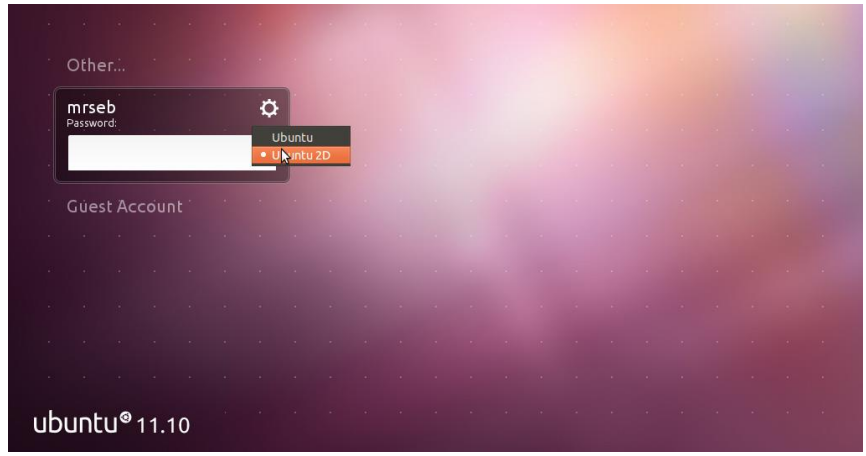
- Vamos usar o sistema operacional Linux (distribuição Ubuntu)
- Ubuntu
  - Código Aberto
  - Construído a partir do núcleo do Linux (baseado no Debian)
  - Patrocinado pela Canonical
  - http://www.ubuntu.com/

ubuntu

Primeiros Passos...

- Boot do computador no Linux
- Quando aparecer a mensagem login colocar usuário e senha cadastrada no NUCC-LAB
- Aparecera o ambiente gráfico do Ubuntu (Servidor gráfico)

Login no Ubuntu



Características do Unix

- Portabilidade: muitas arquiteturas suportadas
- Interoperabilidade em Rede: execução remota
- Multitarefa
- Multiusuário
- Memória virtual
- Diversos programas
- 40 anos de idade (desde 1968)
  - http://en.wikipedia.org/wiki/Image:Unix-history.svg

Sistemas Unix

Linux

- Sistema operacional Moderno e Livre baseado no padrão Unix
- Desenvolvido inicialmente por Linus Torvalds em 1991
- Evoluiu através da colaboração de diversas pessoas, distribuídas pelo mundo, através da Internet
- Executa muitos programas existentes no mundo Unix como os do projeto GNU

Interpretador de Comandos

- Interface entre o usuário e o SO
- Pode ser: linguagem de comandos ou interface gráfica (janelas, ícones e etc…)
- Linux: Shell e servidores gráficos (KDE, GNOME, X Window System e etc…)
- Shell:
  - prompt mostra que o shell aguarda comandos ($_)
  - comando é composto de palavra e zero ou mais argumentos espaço é o separador
  - Comandos para: ajuda, gerenciar arquivos, gerenciar diretórios, manipular texto, gerenciar processos, etc.
  - Diversos shells. Por padrão GNU bash (Bourne Again Shell)

Outros Shell para Linux

- ***ksh*** Korn Shell - o mais usado atualmente.
- ***csh*** C Shell - considerado o mais poderoso, sendo largamente utilizado.
- ***rsh*** Remote Shell - shell remoto.
- ***Rsh*** Restricted Shell - versão restrita do *sh*.
- ***Pdksh*** Public domain Korn Shell - versão de domínio público do *ksh*.
- ***Zsh*** Z Shell - compatível com o *sh*.
- ***Tcsh*** versão padronizada do *csh*.

Principais Diretórios

Diretório	*Conteúdo*
/	Diretório raiz do sistema de arquivos. Igual a C:\ no Windows.
/bin	Arquivos executáveis de comandos essenciais, usados na inicialização do sistema.
/boot	Arquivos estáticos necessários à inicialização do sistema. Onde encontra-se o kernel do sistema.
/dev	Armazena os arquivos dos dispositivos de suporte do sistema como: Discos (HD, CdRom, Disquete), portas de impressoras, portas seriais, etc.
/etc	Guarda os arquivos de configuração do sistema
/lib	Arquivos das bibliotecas essenciais ao sistema
/sbin	Arquivos essenciais para funcionamento do sistema, normalmente somente o super-usuário tem acesso a esses arquivos.
/tmp	Diretório de arquivos temporários.
/usr	Arquivos pertencentes aos usuários. (Programas instalados)
/var	Diretório onde são guardados arquivos variáveis sobre o Sistema.
/home	Local onde se encontram as pastas dos usuários comuns do sistema.

Comandos e Programas Úteis

- Antes de Comecar:
  - Linux é Case Sensitive
  - O Atalho "Ctrl + C" não copia e nem o "Ctrl + v" cola!
  - Sua pasta principal é a /home/<login>

ls

- Lista arquivos/pastas de um diretório

- Principais opções (argumentos):
  - -a: mostra arquivos e pastas ocultas
  - -l: mostrar detalhes dos arquivos e pastas
- Exemplo
  - ls –la
  - Esse comando vai listar as arquivos/pastas do diretório atual do terminal

cd

- Comando para se dirigir para uma pasta
- Exemplo:
  - cd /home/arss/public_html
  - Com esse comando o terminal vai para a pasta desejada.

mkdir

- "Make dir"
- Cria um diretório ou uma cadeia de diretórios
- Principais opções (argumentos):
  - -p : Cria cadeia de diretórios se necessário
- Exemplo:
  - mkdir –p /home/arss/uma/aula/sobre/linux
  - Esse comando vai criar essa pasta

rm

- "remove"
- Remove um arquivo ou pasta
- Principais opções (argumentos):
  - -r : recursivamente, remova todos os arquivos e pastas
- Exemplo
  - rm –r /home/arss/uma/
  - Esse comando vai remover a pasta "uma" e todos os arquivos e pastas que estiverem dentro dela;

cp

- "Copy"
- Copia uma pasta ou um arquivo para um determinado destino
- Sintaxe: cd [args] <origem> <destino>
- Principais opções (argumentos):
  - -r : recursivamente, copia todos os arquivos e pastas para o destino
- Exemplo
  - cp /home/arss/arquivo.zip /home/arss/Desktop/
  - Move o arquivo "arquivo.zip" para o Desktop

mv

- "Move"
- Move um arquivo ou pasta para um destino
- Funciona do mesmo jeito que o "cp"
- Mas também é usado para renomear arquivos.

- Exemplo:
  - mv nome.zip nomeNovo.zip
  - Isso renomea o arquivo para o "nomeNovo.zip"

chmod

- O linux tem um sistema de permissões restrito por default
- Todos os arquivos tem um dono(o "owner")
- Por default apenas o dono tem permissão para alterar arquivos
- O "chmod" pode alterar essas permissões

chmod

- Sintaxe: chmod [args] <alteração na permissão> <arquivo>
- Argumentos
  - São 3 números:
    - Permissão do dono
    - Permissão do grupo do dono
    - Permissão para todos os usuários
  - Quanto maior o número mais poder o usuário vai ter! O maio número é 7;
  - Exemplo
    - Chmod 777 pasta
    - Todos os usuários poderão fazer qualquer coisa nessa pasta

vim

- Um Editor de texto bem simples
- Para poder escrever/alterar um texto é só apertar "insert"
- Para executar algum comando do vim é só apertar "esc" digitar o comando
- Comandos uteis:
  - :q - sair do vim
  - :w - Salvar as alterações do arquivo
  - Usar a ! (exclamação) força a execução de algum comando

vim

- Usando o vim:
  - Sintaxe:
    - vim <nome do arquivo>
    - Se o arquivo não existir, o vim criará um se as alterações no arquivo forem salvas

GCC (GNU Compiler Collection)

- Uma coleção de compiladores
- C/C++, Java, Ada, Pascal, Fortran
- Sintaxe:
  - gcc –o <arquivoExecutavel> <arquivoFonte>
  - Essa é a forma mais simples de se compilar um arquivo
  - E para executar um arquivo no linux é só digitar:
    - ./<arquivoExecutavel>

Obtendo ajuda no Linux

- Duas formas básicas:
  - --help : a maioria dos programas ao receber esse argumento mostram como usar o programa.
  - man <nomePrograma> : mostra o manual do programa, também é muito útil.

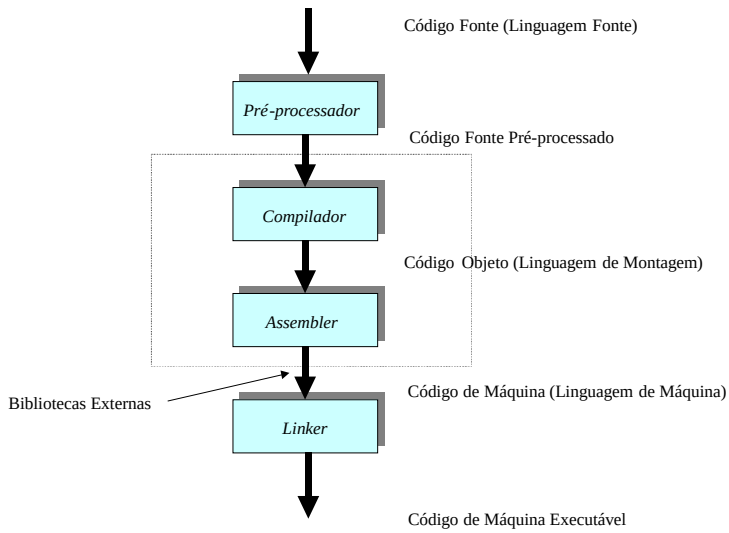
Ambiente de desenvolvimento C

- Como implementar um programa em C?

Criação, compilação e execução

- Os passos de desenvolvimento de um programa em C são geralmente os seguintes:
  - Criação do programa fonte (texto);
  - Compilação desse programa (tradução para código de máquina executável);
  - Execução do código gerado.
- No caso da detecção de qualquer erro em qualquer uma das etapas, todos elas deverão ser repetidas desde o início.

Processo de compilação



Criação de programas

- A criação de programas-fonte em linguagem C faz-se com o auxílio de um editor de texto genérico, ou específico de um ambiente de desenvolvimento.
- No nosso caso, usaremos o sistema operacional Linux como ambiente de desenvolvimento de nossos programas.

Processo de criação de um arquivo para edição do programa:

$touch <nome_do_programa>

Editor de texto: Midnight Commander (mc)

$mc

Compilação

- A compilação dos programas em C faz-se através da invocação de um compilador (por exemplo no Linux, o comando gcc).
- O comando de compilação deverá ser seguido pelo nome do arquivo que contém o código fonte (geralmente com a extensão .c)
- Assim uma compilação básica poderia ser executada no Linux por meio do seguinte comando:

$gcc programa.c

onde "programa.c" é o nome do arquivo contendo o código-fonte.

Compilação

- Se existirem erros de sintaxe no código fonte, o compilador os detectará e indicará a sua localização aproximada juntamente com uma breve descrição da natureza do erro encontrado. Erros na lógica do programa apenas poderão ser detectados durante a execução do mesmo.
- Se o programa não contiver erros de sintaxe o compilador produzirá código de máquina executável. Tratando-se de um programa completo o código executável é colocado em um arquivo chamado "a.out".

Compilação

- Para colocar o resultado da compilação em outro arquivo, que não "a.out", deve-se utilizar a opção "-o" como parâmetro para o compilador. A seguir apresenta-se um exemplo do uso deste recurso:

$gcc -o program program.c

- Neste caso o código executável é colocado no arquivo "program" que é criado com os necessários direitos de execução (no Linux).

Execução

- Se a operação for bem sucedida (compilação sem erros), a execução do programa compilado é feita simplesmente invocando-o como se fosse um comando do sistema operacional:

$program ou
$.\program

Execução

- Durante a execução podem tornar-se evidentes os seguintes erros: erros de execução (p. ex. divisão por zero), ou erros que levem o programa a comportamentos inesperados (p. ex. loop infinito);
- Neste caso, é necessário voltar à edição do programa fonte para corrigir a sua lógica, e depois efetuar também uma nova compilação para produzir a nova versão do código executável.

Exercício 1

Crie, compile e execute o seguinte programa C:

```
void main()  {

    int fahr;
    printf ("\nEntre a temperatura em graus Fahrenheit: ");
    scanf("%d",&fahr);
    printf("\n%d graus Fahrenheit correspondem a graus Celsius %d\n",
    fahr, 5*(fahr-32)/9);

}
```

Exercício 2

Implementar um programa em C que escreva no vídeo o seu nome e endereço, conforme exemplo:

```
$
Marcelo da Silva
Rua Marquês de Paranaguá, 111
Consolação
São Paulo-SP
$
```

Exercício 2 - continuação

a) **Substitua a palavra "main" no programa anterior por "Main". Anote a mensagem de erro emitida pelo compilador.**

b) **Corrija o "Main" (substitua-o por "main"). Retire o fecha parênteses do "main", ou seja, em vez de "main()" digite apenas "main(". Anote a mensagem de erro.**

c) **Corrija o programa e em seguida remova um ponto e virgula qualquer do mesmo. Anote a mensagem de erro. Em qual linha o compilador acusou que havia um erro? Foi nesta linha?**

d) **Corrija o programa e em seguida remova uma aspas " " " do programa. Anote a mensagem de erro gerada. Em qual linha o compilador acusou que havia um erro?**

Exercício 3

Crie, compile e execute o seguinte programa C:

```
void main()  {

    int fahr;
    printf ("\nEntre a temperatura da água em graus Fahrenheit: ");
    scanf("%d",&fahr);
    if (5*(fahr-32)/9 <= 0) printf ("\nEstado sólido\n");

}
```

Exercício 3 - continuação

Modifique o programa anterior para:

```
void main()  {

    int fahr;
    printf ("\nEntre a temperatura da água em graus Fahrenheit: ");
    scanf("%d",&fahr);
    if (5*(fahr-32)/9 <= 0) printf ("\nEstado sólido\n");


}
```

Exercício 3 - continuação

1. **Modifique o programa anterior para determinar em qual dos três estados se encontra a água na temperatura informada:**

   - ✓ **Sólido**
   - ✓ **Líquido**
   - ✓ **Gasoso**

Exercício 4

Crie, compile e execute o seguinte programa C:

```
void main()  {

    int fahr, celsius;
    int inicio, fim, incr;
    inicio=0;
    fim=300;
    incr=20;
    fahr=inicio;
    while (fahr <= fim) {
        celsius=5*(fahr-32)/9;
        printf ("%d\t%d\n", fahr, celsius);
        fahr=fahr+incr;
        }
}
```

Exercício 4 - continuação

1. ***Qual o papel das chaves ("{" e "}") no comando while ?***

3. ***Quantas vezes os comandos que pertencem ao while são executados ? E quantas vezes a expressão do while é avaliada ?***

Exercício 4 - continuação

I. ***Acrescentar um cabeçalho à tabela;***

II. ***Efetuar conversões de 0 a 1000 fahrenheit, de 10 em 10;***

III. ***Inverter as colunas da tabela, apresentando primeiro os graus celsius e depois os graus fahrenheit correspondentes;***

IV. ***Calcular kelvin a partir de graus fahrenheit (C=K-273,15).***

Dúvidas