Universidade Federal de Goiás
Instituto de Informática

Sistemas Operacionais

Trabalho sobre Shell do Linux

ATENÇÃO ESSA É A VERSÃO FINAL

A shell deverá ser desenvolvida por partes ao longo do semestre

Resumo

Você deverá criar uma `shell` -- um interpretador de comandos que lê um comando simples e executa o programa.

Objetivo

O objetivo do trabalho é ajudá-lo a aprender como o processo de manipulação é normalmente realizado em sistemas e familiarizá-lo com programação de sistema (*system programming*). O trabalho também irá ajudá-lo a relembrar alguns aspectos de C.

Trabalho

Escreva um programa em Linux que leia comandos simples do usuário e os execute como as `shells` padrão do Unix fazem. Seu programa deverá permitir uma variedade de modificadores de comandos incluindo redirecionamento de E/S, `pipes`, e processos em `background`.

Seu programa deverá suportar o seguinte:

1. Executar o programa (com ou sem argumentos) no `foreground` ou `background`;
2. Executar o programa (com ou sem argumentos) no `foreground` ou `background` redirecionando a saída para um arquivo. Neste caso, você deverá alterar os descritores de arquivos.
3. Executar o programa (com ou sem argumentos) no `foreground` ou `background` recebendo entrada de um arquivo;
4. Executar o programa (com ou sem argumentos) no `foreground` ou `background` recebendo entrada de um arquivo e redirecionando a saída para outro arquivo;
5. Executar programa1 e programa2 (com ou sem argumentos, usando a saída do programa1 como a entrada do programa2 (isto é um `pipe`). Note que você precisa criar o `pipe` antes de `fork` e não deverá esperar pelo primeiro processo (que precisa estar rodando ao mesmo tempo que o segundo) .
6. Sua shell deverá suportar a variável de ambiente `PATH`. Ou seja, você deverá procurar por aplicações em todos os diretórios dados por `PATH`. Para obter o `PATH`, você poderá usar:

```
char *path;
path = getenv("PATH");
```

7. A implementação de `pipes` talvez seja mais complicado pois você deverá prever erros potenciais, tais como o que acontece se você não consegue iniciar o segundo processo?
8. Você não precisa (por enquanto) criar uma interface particularmente sofisticada. No entanto, você deverá projetar o código de tal forma que seria possível substituir sua interface por uma mais sofisticada. Você deverá reportar erros tais como inabilidade de criar arquivos, impossibilidade de `fork` um processo ou impossibilidade de executar um programa.
9. Você deverá manter uma história dos comandos emitidos pelo usuário, ou seja, uma lista dos 100 últimos comando executados pela `shell`.