MAC0422 - SISTEMAS OPERACIONAIS

Relatório EP1 - Shell

Washington Luiz - 10737157

Ygor Tavela Alves - 10687642

1. Shell

Consiste naturalmente em um loop infinito, em que, é aguardado comandos do usuário. Para ler uma linha de comando, foi criada uma função chamada ***read_command***, usando a função fgets. Com a string de input do usuário em mãos, é realizado uma conversão deste input para uma array de array de chars, usando a função ***parse_command***, cujo resultado será usado pelas chamadas de sistema realizadas durante a execução dos comandos.

Quando concluído este pré-processamento, é verificado se o comando dado está definido no programa (“protegepracaramba”, “liberageral”, “rodeveja” e “rode”) para então executar sua função correspondente e realizar as tarefas necessárias e, caso não esteja definido, o programa retorna uma mensagem de erro. Em todos os comandos, é usado uma chamada de sistema ***fork***, para executá-los como processos filhos do shell. Além disso, todos os comandos fazem uso da chamada _***exit*** para finalizar o processo. Abaixo, descrevemos o funcionamento de cada comando.

2. Comandos

2.1 protegepracaramba

Esse comando deve receber por parâmetro o path do arquivo a ser protegido. Então, o path é passado para a chamada de sistema ***chmod***, que também deve receber o nível de proteção no padrão dos 9 bits de segurança para os tipos de usuário user, group e owner, 0XXX (octal), no caso do comando, é utilizado a numeração 0000.

2.2 liberageral

É análogo ao item 2.1, mas os bits correspondentes são 0777.

2.3 rodeveja

Nesse comando é usado a chamada de sistema ***execve***, no qual é passado por parâmetro o path do programa a ser executado, todos os argumentos, inclusive o programa, e as variáveis de ambientes. Se houver algum erro, o programa é encerrado. Quando esse processo filho encerra, o processo pai retorna o status de saída com o qual o filho terminou.

2.4 rode

Esse comando é análogo ao 2.3, porém ele é executado em background. Sendo assim, é necessário fazer com que os sinais **SIGINT** e **SIGQUIT** sejam ignorados no processo filho e que o processo pai ignore o sinal

SIGCHLD, para isto, fizemos uso da chamada ***signal***. Também é preciso fechar a entrada padrão (a shell monopoliza o teclado), usando a chamada de sistema ***close***.

2.5 exit

Encerra o shell.