EP1 – MAC422 - Sistemas Operacionais

Daniela Gonzalez Favero 10277443
Felipe Castro de Noronha 10737032

Neste primeiro EP, implementamos uma shell rudimentar para o MINIX. Nossa shell pode receber os seguintes comandos:

1. `liberageral <file-path>` : dá permissão 777 á um arquivo
2. `protegepracaramba <file-path>` : dá permissão 000 á um arquivo
3. `rodeveja <file-path>` : executado um arquivo e imprime o status retornado pelo mesmo
4. `rode <file-path>` : executa um arquivo em plano de fundo

1. Implementando a parte externa da shell

A função `main()` tem um laço infinito que recebe o input do usuário. A função `parse_arg()` trata a entrada, separando-a em *comando* e *caminho*, repassando essas duas informações para a função `process()`, que por sua vez chama as funções que executarão as chamadas de sistema.

Fizemos também algumas funções auxiliares, são elas:

- `valid_file()` : Faz a checagem dos arquivos, vê se tal arquivo existe ou se ele é executavel, impedindo que falhas ocorram durante a execução da shell.
- `remove_lf()` : Remove o caractere `\n` do final de uma string.
- `word_count()` : Conta o numero de palavras em uma string. Usada para tratamento de texto inserido na shell.

2. Implementando `protegepracaramba()` e `liberageral()`

Implementadas usando a chamada de sistema `chmod()` para mudar a permissão do arquivo passado, usando os valores retornados pela função para checar se o processo ocorreu normalmente.

3. Implementando `rode()` e `rodeveja()`

Implementadas usando as chamadas de sistema `fork()` para criar um processo filho da shell e `execve()` para executar o arquivo passado. A função `rodeveja()` utiliza também `waitpid()` para aguardar o processo terminar e `WEXITSTATUS()` para traduzir a saída do programa após o processo ser encerrado. Aqui, também, foi feito um controle/validação com todos os codigos retornados por cada chamada de função.