Nome: Giovanna Rodrigues Mendes RGA: 2021.1904.032-7
Nome: Matheus Felipe Alves Durães RGA: 2021.1904.008-4

Professor: Ronaldo Alves Ferreira
Disciplina: Sistemas Operacionais Turma: P01-2023-1

# Relatório

Durante a execução do trabalho, enfrentamos muitos problemas principalmente na parte do analisador léxico e sintático, porque, primeiramente, tentamos implementar do modo como foi apresentado tanto no documento disponibilizado quanto explicado em aula. Contudo, isso não foi possível, pois os computadores do laboratório e dos integrantes não estavam indo de acordo com o esperado. Por isso, decidimos fazer de forma simplificada ambos analisadores na mão.

Apesar de tentarmos seguir os passos do PDF, infelizmente não conseguimos entender direito algumas partes do mesmo, fora que ele encontra-se desatualizado em algumas questões, algumas funções que ele mostra não se comportam mais da mesma maneira, inclusive adquiriram mais parâmetros, sem falar em algumas partes que o PDF parecia estar corrompido, por possuir caracteres irreconhecíveis.

Conseguimos implementar o analisador léxico e sintático, porém com algumas limitações aos comandos, sendo essas, sempre que digitar algum comando, parâmetro ou modificador, ele sempre deve ser separado por um espaço, como no exemplo a seguir: “ls -l | grep .cpp > arquivoDeSaida.txt” .

Ademais, houveram algumas implementações desejadas na descrição do trabalho que foram implementadas, mas não estão funcionando consoante o previsto, como o Crtl+D. De acordo com as nossas pesquisas, o sinal passado pela pressão dessas teclas seria o SIGQUIT, porém não funciona o fechamento da shell ao pressioná-las.

Além disso, alguns outros comandos não foram criados por problemas na construção do código o que gastou muito tempo, por isso decidimos não deixá-las no código final. São eles: bg, fg, jobs, kill e export.

Entretanto, os demais conseguimos construir. Vale salientar que, os comandos colocados executam de forma simples, ou seja:

- cd: pode ser passado tanto .. quanto um nome de um diretório, se este existir, ele avançará o endereço do path para dentro do diretório, caso não exista retornará um erro, caso seja .., voltará o path para a pasta pai da atual;
- history: printa, ao ser chamado, na tela os últimos 50 comandos invocados que são guardados no arquivo .history e lidos no momento da chamada. Vale lembrar que, o arquivo é excluído ao fechar o terminal (exit);
- echo: escreve o que for passado como parâmetro, caso seja passado "$” em um parâmetro, ele irá retirar o “$” e tentar buscar uma variável de ambiente com esse parâmetro;
- pwd: chamando dessa forma, o pwd mostra, no terminal, a variável de ambiente representada pelo path sendo atualizada no cd;
- set: mostra todas as variáveis de ambiente;
- exit: ao ser invocado, desse modo, fecha o arquivo .history, exclui-o e fecha a execução com status 0 (zero).

Vale ressaltar que, quando se trata de redirecionamento de entrada, de saída e de erro, somente os comandos externos possuem, assim como eles também possuem pipes. Já os comandos internos, não possuem nem os pipes nem os redirecionamento de entrada,

de saída e de erro. Isso, por causa de uma falha deste. Vejamos um exemplo a seguir do erro:

```
$tecii: ls -l | grep .cpp
$tecii: … ( Resposta certa )
$tecii: pwd
$tecii: … ( Resposta do pwd )
Use: grep (standart input) [OPTIONS]...
```

Outrossim, teve uma implementação que queríamos fazer, todavia não conseguimos achar a forma certa e se tem como fazer. Essa seria utilizar o ios::ate para a abertura, leitura e escrita do arquivo de histórico. Este modo de abertura seria encarregado de abrir o arquivo no final além de ler e escrever nessa posição. Como o history imprime na tela os últimos 50 comandos (ou menos), nós queríamos pegar o último comando do arquivo e ir subindo o ponteiro de leitura. No entanto, não conseguimos êxito. Por isso, lemos o arquivo do início ao fim, printando apenas os últimos 50 comandos.

Infelizmente, percebemos de última hora que no relatório dizia que o >> seria usado para saída de erro, nós estávamos usando >> para saída ao final do arquivo passado, tivemos que mudar a estrutura do código inteira por conta disso.

O erro relatado mais acima foi feito em uma versão do código, que infelizmente se perdeu durante o desenvolvimento. Tentamos replicá-la para mandar junto em outro arquivo, mas não conseguimos.

Para a execução do makefile pode ser passado os comandos:

- make gcc : para compilar o trabalho apenas;
- make shell :  compilar e executar;
- make clean : remove o arquivo executável.