Relatório da Atividade Prática 2: Processos e Shell

Laila Pereira Mota[1], **Matheus Hofstede**[1], **Rafael Correa Nagy**[1]

[1]Instituto de Matemática e Estatística – Departamento de Ciência da Computação
Universidade Federal da Bahia (UFBA)
Salvador – BA – Brasil

1. Considerações gerais

O objetivo deste projeto é a realização da implementação de um programa que simula o shell do sistema operacional de forma que a equipe possa aprender o relacionamento entre o kernel, o shell e os programas de usuário, aprender como usar as syscalls do Unix-like para gerenciamento de processos, bem como ganhar mais experiência na no entendimento e manipulação de erros do sistema.
Como os membros da equipe já são familiarizados com lógica de programação, com a linguagem C e com a utilização do git, optou-se por utilizar um repositório git para gestão, controle de versão e trabalho colaborativo. Não foram necessárias grandes revisões em outros aspectos.
A equipe realizou um estudo sobre as chamadas de sistema e bibliotecas fork, execvp, wait, witpid, kill, exit, signal, printf, fgets, strtok, strcmp, strsignal, atoi, chdir, getwc e os erros provenientes de tais chamadas:

- Para entender as chamadas, foi utilizada a documentação dos manuais do linux (sugeridos pela atividade) que foi suficiente para utilizar os métodos com seus parâmetros e structs corretamente, mas, quando necessário, foram realizadas consultas ao stackoverflow e outros materiais disponíveis na internet para resolver erros.
- Para lidar com os erros, foi utilizado o header do sistema errno.h.
- Para modularização do código, de forma a torná-lo mais legível, foi criado o arquivo commands.c com todas as chamadas de funções requisitadas pela atividade.

O repositório está disponível em https://github.com/hofstede-matheus/so-ufba-lab-02-myshell, onde é possível encontrar as orientações e comandos de execução, descritas no ”README.md”, a partir do makefile, com as seguintes informações:

```
MYSHELL
Esse projeto faz parte do Estudo Dirigido 02
Clone o projeto, e no diretório raiz, execute em uma linha de comando:
make myshell
Os arquivos gerados estão na pasta build/
```

2. Exercício: Implementando myshell

Para esse exercício, adicionamos uma verificação inicial dos parâmetros fornecidos. A entrada é dividida em tokens, que são passados, através da função invoke, para a sua execução correspondente, seguindo as orientações da atividade. As funções implementadas foram: start, wait, waitfor, run, watchdog, chdir, pwd, quit/exit.
O programa shell identifica entradas inválidas e retorna uma mensagem, informando o

comando inexistente.
Para que as mensagens de erros fossem retornadas de maneira amigável, criamos uma biblioteca auxiliar que chamamos de “errors”. Ela contém funções que irão sobrescrever as funções de erros padrão do C, essas funções consistem apenas em uma impressão de mensagens referentes ao erro, e depois, retorna a falha para o sistema. Além disso, também existe um header para essa biblioteca.