Trabalho 3 (Fases 6 e 7)

Edson Ricardo da Costa, Jonatas Van Groll Lemos e Lourenço Souza

¹ Sistemas Operacionais - Turma 031

² Graduação de Engenharia de Software – Escola Politécnica

³ Pontifícia Universidade Católica do Rio Grande do Sul(PUCRS)

{Edson.Costa, Jonatas.Lemos, Lourenco.Souza} @edu.pucrs.br;

Resumo. *Este pequeno resumo descreve como foi desenvolvido o terceiro trabalho da disciplina de Sistemas Operacionais, no 2º semestre de 2021. Mostraremos como foi feita a implementação do trabalho explicando o que foi desenvolvido, como o gerenciador IO e o terminal concorrente, como se deve executar o programa para poder testar os casos de teste e mostraremos também o resultado dos casos de teste desenvolvidos.*

DESENVOLVIMENTO DA SOLUÇÃO

Com intuito de deixar a solução mais parecida com o código base do professor, para a entrega da fase 6 e 7 nós continuamos utilizando o arquivo "*sistema.java",* nele estão implementados todos os itens da parte 1, 2, 3, 4, 5, 6 e 7. Como a fase 6 e 7 ainda era de implementação pequena, o grupo manteve em um único arquivo o sistema, todo o código desenvolvido segue o padrão sugerido pelo professor, onde temos um código completo e detalhado com comentários. Seguindo essa abordagem, o grupo teve vantagens em desenvolver a nova parte proposta, pois o código já era conhecido pelos integrantes, o que facilitou a implementação do gerenciador de memória e o gerenciador de processos.

A proposta do enunciado da fase 6 era desenvolver um IO concorrente, esse IO seria responsável por controlar o sistema e coordenar o menu e o trap handler. Além disso, diferentemente da fase 5, o sistema atual deveria ter um estado extra, agora o sistema contaria com três estados (Running, Ready e Blocked), permitindo assim, outros programas executarem enquanto alguém estivesse bloqueado aguardando alguma resposta do sistema, além desses estados, para manter a funcionalidade e saber quando um processo finalizou, o grupo implementou um quarto estado chamado "*Finished*".

Além do IO Concorrente, outro ponto importante era o shell concorrente, sendo assim, foi implementado um terminal onde poderíamos executar todas as funcionalidades do sistema, além disso, quando inserimos um programa que tenha uma chamada de sistema, seja de input ou output, o sistema deve lidar corretamente com isso e permitir que outros programas continuam rodando enquanto o programa fica no estado de blocked, no aguardo do input do usuário.

Falando dessa forma, toda a implementação parece algo simples, porém tivemos que alterar diversos pedaços de código para poder suportar as duas novas fases do sistema, que compõem o sistema operacional completo, além disso tivemos que desenvolver uma nova interrupção e toda uma nova rotina para quando ocorresse ela.

Com isso, criamos um sistema capaz de atender todos os requisitos do enunciado da fase 6 e 7.

RODANDO O PROGRAMA

Para executar o sistema e poder realizar os testes, primeiro você precisa localizar a função "*main*" escrita no arquivo "*Sistema.java*". É por ela que você executará todos os testes criados pelo grupo, essa função é responsável por controlar o sistema, ou seja, no início da função nós criamos uma nova instância do sistema e logo abaixo chamamos todos os seus respectivos testes, porém diferente das outras versões do sistema, essa versão conta com um menu concorrente, então para testar o sistema você precisa apenas iniciar uma vez o sistema, então para realizar isso apenas rode a função "*main*"e utilize os comandos do menu para navegar e testar o sistema.

Além da instância do sistema, nós também temos uma variável chamada "*enableProcessMenagerWarnings*", essa variável é utilizada como uma flag, ou seja, uma variável de controle para sabermos se queremos ver ou não os avisos que o gerenciador de processos exibe para nós durante a execução do sistema.

Olhando para a imagem abaixo fica mais claro de entender, para rodar o sistema, execute a função main, nela nós temos uma instância do sistema e temos também a flag de warnings definida para false, caso você queira ver os avisos, apenas habilite a flag. Abaixo temos o sistema inicializado, ele chama o menu e logo em seguida, após rodar, você verá na sua tela um terminal com o menu exibido na imagem abaixo, o menu é intuitivo e fácil de utilizar.

```java
Run | Debug
public static void main(String args[]) {
    //cria uma nova instancia do sistema
    Sistema s = new Sistema();
    //ajusta a flag para o gerenciador de processos exibir mensagens ou nao
    enableProcessMenagerWarnings = false;

    //test1 - programa que testa fibonnaci
    //s.test1();
    //test2 - programa que testa progminimo
    //s.test2();
    //teste3 - programa que testa fatorial
    //s.test3();
    //test4 - programa que testa interrupções de endereço invalido
    //s.test4();
    //teste5 - programa que testa manipulador de chamada de sistema(trap 1 - input)
    //s.test5();
    //teste6 - programa que testa manipulador de chamada de sistema(trap 2 - output)
    //s.test6();
    //test7 - programa que testa interrupções de intrução invalida
    //s.test7();
    //teste8 - programa que testa interrupções de overflow de operações matematicas
    //s.test8();
    //teste9 - Programa que testa o gerenciador de memória, esse programa apenas carrega a memória com dados "inuteis"
    //s.test9();
    //test10 - programa que testa o gerenciador de processos, esse programa carrega cinco fatoriais diferentes
    //s.test10();
    //test11 - programa que testa o gerenciador de processos caso ocorra alguma interrupção que quebre o sistema
    //s.test11();

    s.menu();      You, a few seconds ago • Uncommitted changes
}
```

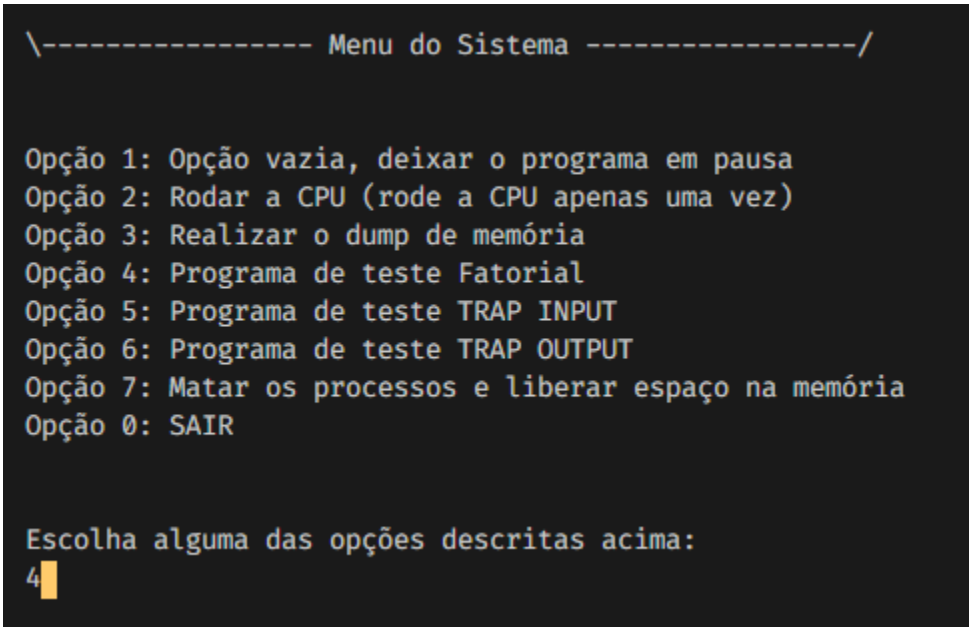
```
\----------------- Menu do Sistema -----------------/


Opção 1: Opção vazia, deixar o programa em pausa
Opção 2: Rodar a CPU (rode a CPU apenas uma vez)
Opção 3: Realizar o dump de memória
Opção 4: Programa de teste Fatorial
Opção 5: Programa de teste TRAP INPUT
Opção 6: Programa de teste TRAP OUTPUT
Opção 7: Matar os processos e liberar espaço na memória
Opção 0: SAIR


Escolha alguma das opções descritas acima:
```

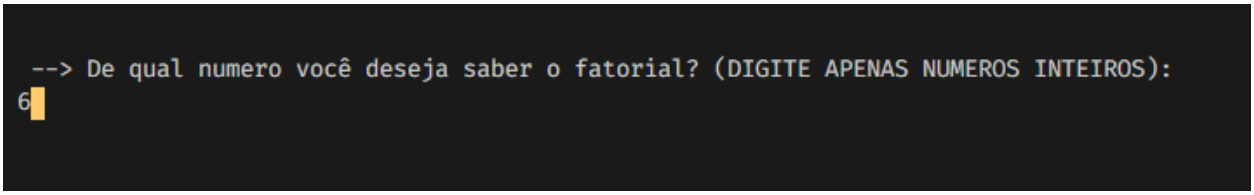
Tentamos deixar o menu o mais simples possível, para que seja fácil visualizar as opções e compreender o que o sistema rodou ou não rodou durante a execução. Para compreender melhor a implementação ou tentar alterar algo, acesse o código disponibilizado na entrega do trabalho, tudo implementado está devidamente documentado no código, tentamos deixar tudo bem claro para que uma pessoa que não conheça muito o trabalho consiga abrir o código e entender o que cada linha do arquivo deve fazer.

RESULTADO DOS TESTES

Este tópico é destinado apenas a apresentação dos resultados obtidos na execução de cada um dos testes desenvolvidos, para compreender melhor o código você precisará abrir o "*Sistema.java*" e analisar como montamos cada um dos códigos.

Como atualmente temos um menu concorrente, mostraremos um teste end to end, ou seja, do início ao fim. Iniciaremos o sistema, colocaremos alguns programas e rodaremos o sistema para executar os testes. Abaixo, observe as imagens para entender o passo a passo que foi executado.





```
 --> De qual numero você deseja saber o fatorial? (DIGITE APENAS NUMEROS INTEIROS):
5
```

```
--------------------------------------------------------------------------------------------
Dump de memória com os programas no sistema:
0:    [ LDI, 0, -1, 6  ]
1:    [ LDI, 1, -1, 1  ]
2:    [ LDI, 6, -1, 1  ]
3:    [ LDI, 7, -1, 8  ]
4:    [ JMPIE, 7, 0, 0  ]
5:    [ MULT, 1, 0, -1  ]
6:    [ SUB, 0, 6, -1  ]
7:    [ JMP, -1, -1, 4  ]
8:    [ STD, 1, -1, 10  ]
9:    [ STOP, -1, -1, -1  ]
10:   [ DATA, -1, -1, 720  ]
11:   [ ___, -1, -1, -1  ]
12:   [ ___, -1, -1, -1  ]
13:   [ ___, -1, -1, -1  ]
14:   [ ___, -1, -1, -1  ]
15:   [ ___, -1, -1, -1  ]
16:   [ LDI, 0, -1, 5  ]
17:   [ LDI, 1, -1, 1  ]
18:   [ LDI, 6, -1, 1  ]
19:   [ LDI, 7, -1, 8  ]
20:   [ JMPIE, 7, 0, 0  ]
21:   [ MULT, 1, 0, -1  ]
22:   [ SUB, 0, 6, -1  ]
23:   [ JMP, -1, -1, 4  ]
24:   [ STD, 1, -1, 10  ]
25:   [ STOP, -1, -1, -1  ]
26:   [ DATA, -1, -1, 120  ]
27:   [ ___, -1, -1, -1  ]
28:   [ ___, -1, -1, -1  ]
29:   [ ___, -1, -1, -1  ]
30:   [ ___, -1, -1, -1  ]
```

Acima podemos ver na primeira imagem o menu concorrente onde podemos fazer um dump de memória, realizar a adição de programas, rodar os programas e finalizar o sistema operacional. Depois disso adicionamos vários programas que resolvem o fatorial de algum número, após fazer isso iniciamos e rodamos todos os programas, devemos fazer isso apenas uma vez, para poder utilizar o sistema operacional novamente nós precisamos iniciar e rodar novamente a função “main”. Após rodar os programas nós podemos selecionar a opção de dump de memória, com ela nós podemos ver que todos os programas foram rodados e o sistema completou o seu ciclo

de execução. Podemos visualizar o resultado do fatorial na posição 10 e na posição 26, respectivamente o fatorial de 6 e depois o de 5, na mesma ordem que foram adicionados.