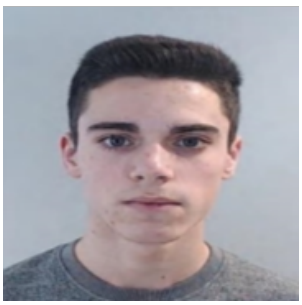
Universidade do Minho
Escola de Engenharia

Sistemas Operativos

Serviço de monitorização de programas executados
GRUPO 44

João Pedro Machado Ribeiro A95719
Telmo José Pereira Maciel A96569
Nuno Miguel Leite da Costa A96897

A97327



A95454

A96897

13 de maio de 2023

Índice

1 Introdução

Na Unidade Curricular de Sistemas Operativos, recebemos o desafio de desenvolver um projeto de modo a colocar-nos à prova perante maneiras de pensar e um conjunto de system calls presentes nos sistemas UNIX. O projeto consiste na implementação de um serviço que permite a monitorização de programas executados numa máquina. Durante todo o processo de desenvolvimento, enfrentamos o desafio de implementar as funcionalidades específicas exigidas no enunciado do projeto, ao mesmo tempo em que garantimos o funcionamento adequado de todas as outras operações.

2 Arquitetura do Programa

A aplicação é composta por dois programas principais: o servidor "monitor.c"e o cliente "tracer.c". Estes programas comunicam entre si através o dois FIFOS (denominados de pipes com nome), um no sentido servidor/cliente e outro no sentido cliente/servidor.

O servidor "monitor.c"tem a responsabilidade de manter em memória a informação dos processos em execução e em ficheiro a informação das execuções dos processos que já foram realizados. Além disso, o servidor recebe notificações do cliente antes e depois, de se processar uma execução e também é responsável pela funcionalidade das estatísticas.

Por outro lado, o cliente "tracer.c"tem a capacidade de notificar servidor (referido em cima), tal como executar programas dos utilizadores, permitindo também notificar o utilizador através do standard output.

Esta abordagem permite uma comunicação eficiente e flexível entre o cliente e o servidor, permitindo a execução de comandos e o processamento dos mesmos pelo servidor. O uso de FIFOS simplifica a comunicação entre os dois programas, facilitando a troca de informações.

3 Funcionalidades Implementadas

3.1 Struct EXEC e Lista Ligada de EXECs

Para este projeto, decidimos começar por criar uma estrutura (struct) que representasse uma execução. Essa estrutura possui os seguintes atributos: PID do processo no qual a execução ocorreu, nome da execução e duração da mesma. Em seguida, pensando na funcionalidade do status (que será abordada posteriormente), prosseguimos com a criação de uma lista ligada de estruturas mencionadas anteriormente. Nessa lista, seriam mantidas as execuções que ainda estivessem em execução, facilitando assim o controlo e o acesso a elas.

3.2 Execução Básica (execute -u)

Nessa funcionalidade, inicialmente enviamos as informações do cliente para o servidor informando que a execução ainda não começou. Em seguida, enviamos as informações do PID, nome e timestamp. Ao receber essas informações, o monitor cria uma execução e a adiciona a uma lista encadeada. Mais tarde,

o processo pai informa que o processo filho pode começar a execução. Quando a execução é concluída, o cliente informa novamente o monitor que a execução acabou, enviando o PID, nome e timestamp. Ao receber essas informações, o monitor remove a execução da lista encadeada e cria um arquivo com as respectivas informações da execução.

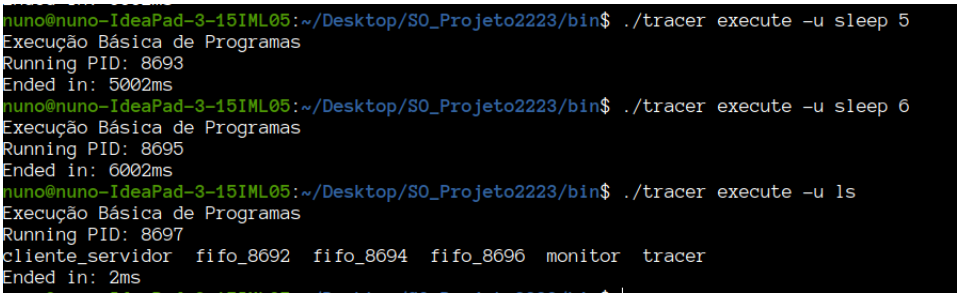


Figura 1: Exemplo do execute -u

3.3 Status

Para executar a funcionalidade de status, o cliente envia ao servidor o nome do FIFO de escrita. O servidor armazena esse nome e, em seguida, chama a função responsável por executar o status. Essa função é executada dentro de um processo filho, que recebe como argumentos uma lista ligada, que é iniciada durante a inicialização do servidor, e o nome do FIFO.

Dentro da função, o FIFO é aberto para escrita e, em seguida, é verificado o tamanho da lista ligada. Esse tamanho é enviado ao cliente. Se a lista ligada estiver vazia, é enviado ao cliente uma mensagem a informar que não existem programas em execução. Caso contrário, todas as execuções em andamento são enviadas ao cliente. Ao receber essas informações, o cliente as escreve no stdout para informar o utilizador.

Figura 2: Exemplo do status

3.4 Stats-Time

Para executar a funcionalidade de stats-time, o servidor recebe do cliente a solicitação para executar o stats-time. Em seguida, o servidor recebe o nome do FIFO de escrita, tal como os nomes dos arquivos (PIDs) que serão utilizados, um de cada vez. O servidor armazena esses nomes em um array de

strings. Posteriormente, o servidor chama a função responsável por processar o stats-time, passando como argumento o array de strings, o número de strings presente nele, o caminho para a pasta onde os arquivos estão localizados e o nome do FIFO de escrita. Dentro dessa função, é aberto o fifo para escrita e é acedido cada arquivo cujo nome está no array de strings, e é acedido o tempo de execução de cada um deles, somando esses tempos a um contador. No final, a soma do tempo de execução de todos os arquivos passados como argumento no início é enviada ao cliente. Ao receber essas informações, o cliente as escreve no stdout para informar o utilizador.

Figura 3: Exemplo do stats-time para o execute -u

3.5 Stats-Command

Na funcionalidade do stats-command, tudo é semelhante ao comando stats-time, exceto pela parte em que chamamos a função responsável por executar o stats-command. No array de strings, na posição [0], temos o nome do programa que desejamos verificar quantas vezes foi executado. A função responsável por processar o stats-command recebe os mesmos argumentos que o comando stats-time. Dentro dessa função, o fifo é aberto para escrita e percorremos cada arquivo e verificamos se o nome do programa corresponde ao nome passado no início. Sempre que houver uma correspondência, incrementamos um contador. No final, esse contador é enviado de volta ao cliente. Ao receber essas informações, o cliente as escreve no stdout para informar o utilizador.

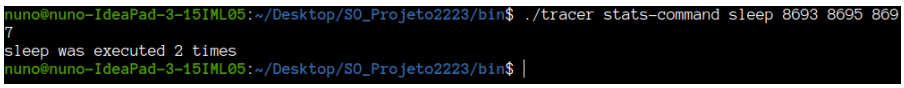


Figura 4: Exemplo do stats-command para o execute -u

3.6 Stats-Uniq

Tal como o comando stats-command, o comando stats-uniq"segue a mesma estrutura inicial do comando "stats-time". Portanto, o início desses comandos é idêntico, diferindo apenas na função chamada para executar o "stats-uniq". Os argumentos passados são os mesmos mencionados anteriormente para ambos os comandos "stats". Dentro da função, o FIFO é aberto para escrita e um array é criado para armazenar todas as primeiras ocorrências dos nomes dos programas, evitando repetições. No final, enviamos cada elemento desse array de volta ao cliente. Ao receber essas informações, o cliente as escreve no stdout para informar o utilizador.

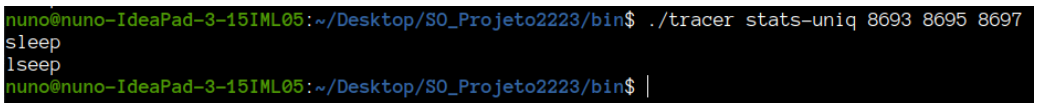


Figura 5: Exemplo do stats-uniq para o execute -u

3.7 Criação da pasta com os ficheiros com a informação das execuções acabadas

Para esta funcionalidade, inicialmente pensamos em criar a pasta com o nome do argumento que era passado durante a inicialização do servidor. No entanto, decidimos criar a pasta no makefile, sempre com o mesmo nome (PIDS-FOLDER), e posteriormente passamos esse nome no ./monitor. Após a conclusão de uma execução, criamos um arquivo com o nome do PID em que foi executado e dentro dele salvamos o nome do programa, bem como o tempo total de execução.

3.8 Informação nos ficheiros

A informação sobre as execuções terminadas, encontra-se nos ficheiros da seguinte maneira:

```
1 sleep
2 2003
```

Figura 6: Exemplo da informação sobre as execuções

4 Conclusão e Análise Critica

Durante o desenvolvimento deste projeto, enfrentamos dificuldades na comunicação entre o tracer e o monitor, resultando em problemas para enviar e ler as informações corretamente,que depois conseguimos resolver. Além disso, enfrentamos desafios na implementação da execução por pipeline, e acabamos por não conseguir concluir essa tarefa.

Em suma, este projeto da Unidade Curricular de Sistemas Operativos nos permitiu aplicar todo o conhecimento que adquirimos durante as aulas, bem como desenvolver habilidades de pensamento crítico e solução de problemas. Foi uma oportunidade para colocarmos em prática os conceitos aprendidos.