Sistemas Operativos – Licenciatura em Engenharia Informática

Rastreamento e Monitorização de Execução de Programas

Grupo 63 – João Pedro Baptista a100705
João Pedro Rodrigues a100896
Rafael Peixoto a100754

13/05/2023

Índice

Resumo

Realizou-se este projeto no âmbito da cadeira de Sistemas Operativos do curso de Licenciatura em Engenharia Informática na Universidade do Minho.

O trabalho consiste num sistema de monitorização de programas executados numa máquina. Tal sistema permitirá ao utilizador executar programas através do cliente, enquanto o servidor guarda as informações dos mesmos, permitindo ao Administrador de Sistemas pedir certas informações dos programas que estão a decorrer e aos que já terminaram.

Programa Monitor

O monitor (servidor) começa por criar dois FIFO's: um para leitura, outro para escrita. O programa espera por conseguir ler algo do FIFO de leitura (i.e. quando o programa tracer escreve algo no seu FIFO de escrita).

Apenas existem 5 comandos que o servidor pode receber no seu FIFO de leitura:

· executing: recebe informações sobre um processo em execução e armazena essas informações em uma estrutura de dados;

· executed: recebe informações sobre um processo que terminou de ser executado e armazena essas informações em outra estrutura de dados;

· status: retorna para o programa tracer informações sobre todos os processos em execução, incluindo o PID, nome do programa e tempo de execução;

· stats_time: recebe uma lista de PID's e retorna para o programa tracer o tempo total de execução desses processos;

· exit: encerra o programa.

O programa utiliza duas estruturas de dados diferentes para armazenar informações sobre os processos. A primeira estrutura, "executing_process", armazena informações sobre processos que estão em execução, incluindo o PID, nome do programa, e o tempo de início da execução (em segundos e microssegundos). A segunda estrutura, "executed_process", armazena informações sobre processos que já terminaram a execução, incluindo o PID, nome do programa e o tempo de execução já calculado (em milissegundos). Para guardar esses dados, o servidor utiliza duas arrays dinâmicas: uma para os programas em execução no momento, e outra para os programas já terminados.

O programa utiliza um loop para aguardar por comandos vindos do pipe de leitura. Quando um comando é recebido, o programa utiliza a função "strtok" para separar os diferentes argumentos do comando. Depois, o programa executa a ação correspondente ao comando. Para os comandos "executing" e "executed", o programa verifica se há espaço suficiente nas estruturas de dados para armazenar as informações sobre o processo. Se não houver espaço suficiente, o programa realoca mais memória para as estruturas de dados.

No final, liberta a memória alocada para as arrays de processos terminados e processos a decorrer.

Programa Tracer

O programa "tracer" monitora a execução de um programa especificado pelo utilizador e envia informações sobre o programa para o pipe de leitura do programa "monitor". O programa "tracer" cria um processo filho que executa o programa a ser monitorado, enquanto o processo pai fica à espera a execução do programa. Quando o processo filho faz uma chamada ao sistema exec, o programa "tracer" envia informações sobre o processo para o pipe de leitura do servidor. As informações enviadas incluem o PID do processo, o nome do programa e o tempo de início da execução (em segundos e microssegundos). Quando o processo filho termina a execução, o programa "tracer" envia informações de terminação do programa para o pipe de leitura do programa "monitor". As informações enviadas incluem o PID do processo, o nome do programa e o tempo de execução (em milissegundos).

execute_u

A função execute_u recebe uma string que representa um programa para ser executado e um inteiro que representa um FIFO de escrita. A função divide a string do comando em tokens separados por espaços em branco e cria um novo processo usando fork(). O novo processo executa o comando usando execvp() e imprime o PID do processo e a string do comando. Se houver um erro na execução do comando, o processo filho encerra com EXIT_FAILURE e uma mensagem de erro é impressa.

execute_p

A função execute_p recebe uma string que contém vários comandos separados por um caracter '|'. A função usa strtok() para dividir a string em comandos separados e chama execute_u para executar cada um dos comandos.

status

A função status envia uma mensagem "status;" para o FIFO de escrita e lê as mensagens recebidas do FIFO de leitura até que a mensagem "finished" seja lida. Esta função é usada para imprimir as mensagens de status enviadas por outros processos.

stats_time

A função stats_time envia uma mensagem "stats_time;" seguida pelos argumentos para o FIFO de escrita e lê a resposta do FIFO de leitura. A resposta é a soma de todos os tempos de execução dos processos em milissegundos. A função imprime essa resposta.

Além disso, há uma função strtrim que remove espaços em branco à esquerda e à direita de uma string.

O programa usa vários constantes definidas com #define, como BUFSIZE.

No final, o programa tracer calcula o tempo de execução do mesmo e imprime-o no ecrã.