Unidade Curricular de Sistemas Operativos

Ano letivo 2020/2021

Tiago Guimarães A91689
Bruno Silva A91649
Gonçalo Rodrigues A91641

Junho 2021

Introdução:

Aurras: O binário aurras é o cliente, tem como função enviar os pedidos para o servidor e fá-lo através de um pipe com nome criado pelo servidor.

Aurrasd: O binário aurrasd é o servidor, tem como função ler do pipe com nome os pedidos enviados pelo cliente.

Transform: O binário transform é o que contém a função para aplicar filtros, vai ser chamado pelo servidor através de um exec.

Desenvolvimento:

Aurras:

O cliente caso seja executado o comando ./aurras sem argumentos imprime as formas de utilização do comando.

Caso não seja invocado sem argumentos abre o extremo de escrita do pipe com nome Cliente->Servidor e após contar quantos bytes tem os argumentos e após os concatenar todos escreve a string resultante no pipe com nome para que o servidor a consiga ler.

Após escrever a string vai verificar se o pedido é de status, se for vai abrir o pipe com nome fifoStatus, se não for vai abrir o pipe com nome fifoSC(server->client).

Aurrasd:

O servidor foi feito de maneira a que a main apenas carregue as configurações dos filtros para uma estrutura através da função loadConf, associe as rotinas de tratamento de sinais, crie todos os pipes com nome que vão ser usados e abra ambos os extremos do fifo Cliente -> Servidor através da função setup e depois fique a ler do pipe Cliente -> Servidor até que este seja fechado.

Sempre que o servidor conseguir ler alguma coisa do fifo vai passar essa informação para a função parse_entry que vai separar a string recebida num array de strings e acrescentar o path para o ficheiro de configuração, o path das pastas do filtro e acabar o array de strings com um NULL para ficar pronto para dar exec. Após tratar do array de strings vai invocar a função status ou transform consoante a primeira string do vetor.

Caso seja invocada a função status o servidor vai dar fork e dentro do processo filho que acabou de criar abrir o extremo de escrita do fifoStatus e escrever no fifoStatus as informações que foram previamente carregadas na estrutura de configuração pelo loadConf e depois fechar o extremo de escrita para que o cliente feche após imprimir o status.

Caso seja invocada a função transform o servidor vai dar um primeiro fork que vai servir de monitor e após abrir o extremo de escrita do fifo Server -> Client vai dar outro fork dentro do qual vai correr o binário transform através de um exec, o primeiro fork entretanto

fica à espera que este exec acabe para poder fechar o extremo de escrita do fifo Server -> Cliente fazendo com que o cliente feche sinalizando o final da execução do comando.

O servidor tem dois handlers de sinais, um para o SIGINT e outro para o SIGTERM, caso receba um SIGINT vai apenas enviar um SIGTERM para si próprio e quando recebe o SIGTERM chama a função shutdown que fecha os dois extremos do fifo Client -> Server e fecha os 3 pipes com nome criados pela função setup.

O servidor contém ainda mais 3 funções, incrementInstances, decrementInstances e readln.

A readln é usada pela loadConf para ler o ficheiro de configurações.

A incrementInstances recebe um número de argumentos e um array de strings e para cada filtro que encontrar nesse array de strings vai incrementar a na sua estrutura de configurações a variável current (que indica quantos filtros estão a ser usados).

A decrementInstances recebe o mesmo que a incrementInstances e faz o contrario, para cada filtro que encontrar no array de strings vai decrementar a sua variável current.

Transform:

O transform foi feito de maneira a ler o ficheiro de configurações e por numa estrutura o nome do filtro e o nome do seu binário de execução, após fazer isto chama a função applyFilters.

A função applyFilters recebe o número de argumentos e um array de strings, caso seja só um filtro não vai criar nenhum pipe, mas caso seja mais que um vai criar pipes para os filtros intermediários. Após isso vai executar um fork para cada filtro e dentro de cada um dos forks vai redirecionar o standardInput e standardOutput da maneira pretendida para poder dar exec ao seu devido filtro.

Conclusão:

Apesar de infelizmente não termos implementado todas as funcionalidades pedidas no enunciado, este trabalho foi um meio para melhor entendermos o ambiente linux e a comunicação entre processos através de pipes com nome e pipes anônimos e a monitorização destes.