Sistemas Operativos
Grupo 69

Diana Costa (A78985) Paulo Mendes (A78203)
Pedro Fernandes (A77377)

02 de Junho de 2017

Resumo

Perante a proposta de criar um sistema de *Stream Processing* que permitisse a implementação de uma rede de componentes para filtrar, modificar e processar um fluxo de eventos, houve um impasse inicial devido à necessidade de uma boa estruturação do problema. Tudo isto requeriu a escolha acertada das chamadas ao sistema a utilizar, como *pipes* (anónimos ou não), *forks*, entre outros, de modo a que a implementação deste sistema fosse o mais eficiente e simples possível.

Depois de algum tempo e trabalho, o resultado encontrado foi eficiente e satisfatório, e os objetivos e respostas às questões do enunciado proposto maioritariamente cumpridos.

Conteúdo

1 Introdução

O projeto *Stream Processing* surge no âmbito da unidade curricular Sistemas Operativos, do Mestrado Integrado em Engenharia Informática da Universidade do Minho. Sendo um projeto de pequena/média dimensão a ser resolvido na linguagem *C*, este baseia-se na construção de um sistema de componentes, ou rede, que seja responsável por filtrar, modificar e processar um fluxo de eventos.

Este sistema é composto por componentes, isto é, por operações possíveis a efetuar sobre um conjunto de dados. Estão disponíveis as operações const (programa que reproduz as linhas, acrescentando uma nova coluna sempre com o mesmo valor), filter (programa que reproduz as linhas que satisfazem uma condição indicada nos seus argumentos), window (programa que reproduz todas as linhas, acrescentando-lhe uma nova coluna com o resultado de uma operação calculada sobre os valores da coluna indicada nas linhas anteriores), spawn (reproduz todas as linhas, executando o comando indicado uma vez para cada uma delas, e acrescentando uma nova coluna com o respetivo exit status), e também cut, grep e tee.

Toda esta rede precisa de um controlador, que tem como objetivo a implementação efetiva da rede de processamento de *streams*. Este controlador recebe comandos, ou um ficheiro de configuração, e processa a informação. Assim sendo, um controlador deve ser responsável pela implementação de funcionalidades básicas, entre elas um comando node (que permita definir um nó da rede de processamento), connect (que define ligações entre nós), disconnect (que desfaz a ligação entre nós) e inject (permite injetar na entrada do nó a saída produzida por um comando executado e uma lista de argumentos).

Assim sendo, houve a necessidade de pensar em estruturas e algoritmos para resolução dos métodos, explorando as potencialidades da linguagem *C* e das *System Calls* aprendidas nas aulas práticas de Sistemas Operativos.

Em suma, a Secção 2 descreve o problema a resolver, enquanto que a Secção 3 apresenta e discute a solução proposta pelo grupo, desenvolvida ao longo do semestre. O relatório termina com conclusões na Secção 4, onde é apresentada uma análise crítica dos resultados obtidos. Além disso, é feito um balanço tendo em conta as dificuldades ao longo do desenvolvimento do projeto.

2 Descrição do Problema

No contexto do projeto prático de Sistemas Operativos, pode-se afirmar que foram propostas (ou necessárias) três tarefas. Como primeira instância, seria necessário estudar, ou planear, toda a estrutura da rede. De seguida, era preciso definir todas as componentes que a rede poderia executar. Por fim, restava implementar o controlador em si, e garantir que cumpria as funcionalidades básicas requeridas pelos docentes da Unidade Curricular.

A primeira tarefa consistiu na reunião de todos os elementos do grupo e discução a fundo de como seria estruturada a rede, que *System Calls* seriam necessárias, nas vantagens e desvantagens da utilização de, por exemplo, pipes com nome e sem nome, e por fim vizualizar o que seria o futuro desta rede e se seria eficiente.

Numa segunda tarefa (mais simples), foram distribuídas pelos membros do grupo as componentes const, filter, window e spawn, como forma a rentabilizar o tempo, sendo que todos os membros testaram e analisaram todas as componentes.

Já a terceira tarefa envolveu mais trabalho, dado que foi necessário implementar todo um controlador da maneira mais eficiente, para resolver os requisitos do enunciado e testar se estariam a funcionar corretamente.

3 Concepção da Solução

Em grupo, foi proposto que, em conjunto, seriam resolvidas as grandes questões acerca da escolha e implementação do controlador, e dada a natureza das componentes, estas últimas serias distribuídas pelos elementos do grupo.

Procurou-se assim obter uma linha de trabalho clara, eficaz e que permitisse obter os resultados expectáveis.

3.1 Estruturas de Dados

Dada a natureza deste projeto, e tendo em conta a implementação sugerida em grupo, não foram necessárias quaisquer estruturas de dados, não tendo estas impacto quer na eficiência, quer na resolução dos problemas.

3.2 Implementação

Ao longo do projeto, as dificuldades encontradas foram bastantes e diversificadas. Dado como clarificada a questão das estruturas de dados e quais as tarefas do projeto, resta falar da implementação das tarefas em si, necessárias para a resolução do projeto.

Numa primeira instância, e como já fora referido, foi necessária a resolução das componentes. Entre elas, a "const"foi a de resolução mais simples e rápida, uma vez que se baseava "apenas"em acrescentar uma nova coluna, com um valor fornecido como argumento, a todas as linhas do input. Assim sendo, foi necessário ler o input, guardar num *buffer*, e através de funções como "strcpy()"e "strcat()"foi acrescentado o pretendido, e de seguida reproduzido de novo o input. No que toca à filter, a dificuldade já foi um pouco mais acrescida, dado que incluía fazer uma espécie de "parse"a todo o input, e verificar se o conteúdo de uma dada coluna era inferior (entre outras opções) ao de outra. Através das funções auxiliares "elem column()"e "elemIndexInicial()"foi permitido localizar a coluna pretendida e extrair os números, para depois, na função principal, serem escritos para o output caso cumprissem o requisito. Já na função window, era necessário reproduzir todas as linhas, acrescentando uma nova coluna com o resultado de uma operação (média, máximo, mínimo ou suma) calculada sobre os valores da coluna indicada nas linhas ateriores. Mais uma vez, foi preciso fazer "parse"ao input e codificar as operações possíveis. Por fim, a spawn era de cariz idêntico, mas desta vez era para executar um comando especificado e acrescentar no final de cada linha do input o *exit status*. Também aqui foram usadas funções auxiliares no "parse"do input.

Mudando de tema, agora para o controlador, foi necessária uma discução e reflexão acerca da melhor estrutura para a rede. Assim, o primeiro comando - node - envolveu bastante tempo e planeamento por parte dos elementos do grupo. Foi decidido que cada nó da rede de processamento incluiria três processos principais: um para ler, outro para executar a componente, e outro para escrever, ou seja, ler do pipe de entrada, executar o que é lido e escrever no pipe de saída, e esta seria a vida de um nodo. Assim sendo, quando é invocado o comando "node"de criação de um nó na main, cria-se um processo1 (chamar-lhe-emos assim) que será o do nodo. Este processo criará, por sua vez, um processo para leitura do *named pipe* de entrada que ficará a escrever para o processo de execução (através de um pipe anónimo). O processo1 também criará este processo de execução, responsável por executar quer as componentes, quer os comandos do sistema. Este processo de execução apenas lê do pipe anónimo, executa algo, e o resultado da execução é redirecionado para outro pipe anónimo, que interliga, finalmente, o processo de execução com o processo1. O fluxo de informação passará a ser, então : pipe de entrada do processo de leitura - processo de leitura - pipe anónimo (de entrada) do processo de execução - processo de execução - pipe anónimo (de saída) do processo de execução - processo de escrita (processo1) - pipe de saída do processo de escrita. Todo este sistema permite uma permanência da rede e um fluxo de escrita constante e eficaz, e a rede vai sendo construída ao longo do tempo e da inserção de nós. O segundo comando - connect - foi mais simples de implementar, embora exigisse também perspicácia para manter a eficiência e simplicidade do código. Deste modo, decidiu-se que a conecção entre dois nós ou mais seria efetuada através da criação de um processo que ficaria a ler do pipe de saída de um dos nós

e a escrever noutro. Como é permitida a ligação de um nó a vários, na main existe um *loop* que assegura que várias conecções são efetuadas (processos). Existe um registo interno no programa de todas as ligações atuais entre nós e que permite manipular e conectar os nós pretendidos. O objetivo do comando contrário ao anterior - disconnect - seria desfazer a ligação entre dois nós, criada pelo connect. Ora, se o connect cria um processo, na disconnect procedeu-se ao seu oposto, isto é, à terminação de um processo através do envio da mensagem *SIGTERM*. Esta system call permite que o processo termine o seu "trabalho"e depois "se mate", fechando os descritores de ficheiros que tinha associado. O último comando - inject -, a nível de implementação, é similar ao node. Neste comando é necessário escrever na entrada de um nó a componente que pretende executar. Então, cria-se assim um processo filho que executa o pretendido e envia de novo ao pai o resultado da execução, para ser enviado ao nó.

Finalmente, para auxílio quer das componentes, quer do controlador em si, foi criado um header file "auxs"que permite usar a função "read line()"implementada pelo grupo. Também neste ficheiro está a função "parse cmd()"que permite, no controlador, fazer parse ao comando recebido, transformando uma *string* numa *string* de *strings*.

4 Conclusões

Tendo em conta o objetivo do projeto, a criação de um sistema de *Stream Processing* que permitisse filtrar, modificar e processar um fluxo de eventosa foi de facto um projeto interessante e desafiante para todos os elementos do grupo. A implementação deste sistema permitiu não só o desenvolvimento das capacidades de raciocínio e programação dos alunos, mas também foi como um primeiro contacto com o que será o futuro dos mesmos num possível ambiente profissional.

De facto, o maior obstáculo deste projeto, ao contrário das unidades curriculares anteriores, foi o manuseamento de *System Calls* que permitem controlar processos e pipes, e trabalhar com programação de tão "baixo nível"(não no seu sentido conotativo), característica da UC de Sistemas Operativos, em que é necesário controlar e solicitar "serviços"diretamente do núcleo. Assim sendo, não foi fácil conciliar o estudo de diversas *System Calls*, e a obtenção de uma linha de raciocínio clara que tivesse em conta o desenvolvimento de uma "semi-aplicação", que, na vida real, poderia servir para implementar uma rede de processamento, para ser usada num contexto de processamento de grande quantidade de dados por etapas sucessivas. Todo o esforço feito na escolha das estruturas permitiria uma resolução simples e eficaz de todas as funções propostas, baseadas na criação e destruição de processos que pudessem comunicar através de pipes.

Por último, o resultado foi extremamente positivo. A implementação das funcionalidades básicas deste sistema foram conseguidas com sucesso e o trabalho de semanas por fim deu frutos.