Relatório projeto SO

2016/2017 UM
LCC-TP6

André Pereira - a79196
Joaquim Oliveira - a76958
Rúben Sousa - a65447

1 Componentes

1.1 - Const

Quando o programa arranca, o seu trabalho é ler uma linha do stdin, e de seguida armazena-a (com uma pequena alteração, faz append dessa linha de ":<valor>) num buffer auxiliar. Quando há end-of-file, o programa faz write desse buffer no stdout.

1.2 e 1.3 - Filter e Window

Similar ao filter. Lê linha a linha e vai adicionando-a num buffer auxiliar caso satisfaça a condição passada como parâmetro. No final, quando houver end-of-file imprime o buffer no stdout.

1.4 - Spawn

Se houver "$valor" passado como parâmetro, o programa após ler uma linha, copia essa linha para um buffer e "substitui" o "$valor" do parametro pela coluna respetiva da linha lida. Após isto faz execv com os argumentos passados no arranque de spawn. No final de executar (ou não) o processo pedido, adiciona à linha o respetivo código de erro.

2 Controlador

Estruturação da rede:

Existe o processo que inicia a rede, o “controlador”, que lê linhas do stdin e executa os comandos introduzidos (caso sejam válidos da rede). Este processo é responsável por executar um outro processo, o “server”, que faz a gestão de toda a rede, ou seja, controla os nós, sabe quais os nós que estão ligados a outros, sabe os PIDs de todos os processos que compõem a rede e encarrega-se auxiliar cada nó, gerindo cada um e injetando eventos.

O “server” aloca espaço para uma estrutura de dados (grafos), que serve para armazenar todos os fifos dos nós e também guardar, todos os fifos adjacentes a um nó, ou seja, quais os nós que estão ligados a outros.

Quando o “controlador”, recebe um comando “connect <id> <ids...>”, este comunica com o “servidor” através de um FIFO, e assim enquanto o “servidor” lê linhas do seu FIFO, se receber um “connect...”, este irá à estrutura de dados e adiciona, o nó adjacente ao id1 introduzido. Quando receber um “disconnect”, o “server” elimina o id2 adjacente ao id1 no grafo.

Se é introduzido o comado “node...” no “controlador”, este não comunica diretamente ao “servidor”, mas executa mesmo um processo criado por nós com o nome de “node”. Este processo encarrega-se de criar o seu FIFO, abrir o FIFO do “servidor” caso queira comuniar com ele (quando há inject) e também executará o seu comando que é passado como parâmetro.

No “controlador”, se for inserido no stdin, “inject...”, este comunica ao “servidor” através do seu FIFO, e assim este consegue receber a mensagem e sabe que há uma tentativa de injetar eventos (produzidos pelo output do comando passado como argumento). Então ele encarrega-se de ir à estrutura de dados e procurar pelos nós adjacentes ao id que queremos injetar. Caso encontre adjacentes, então é o servidor que escreve no FIFO

de cada nó adjacente as linhas produzidas pelo comando executado anteriormente. O “node” depois de executado o seu comando, envia os seus eventos produzidos ao servidor para que este possa-as reencaminhar para o nós adjacentes àquele que produziu os eventos. Se nao houver adjacentes, não há redirecionamento de output para outros nós, e assim os eventos são escritos no stdout.

Aspetos avançados:

Remoção de nós

Como temos uma estrutura de dados que no fundo define um grafo, torna-se relativamente simples remover os nós e copiar os adjacentes no id que queremos remover para o/os id/ids que estão conectados ao nó que iremos remover. Assim basta encontrar todos os nós adjacentes àquele que iremos remover e guardá-los num array. De seguida, vamos removendo cada nó daquele que queremos matar e adicionando àqueles que conectavam o id que será removido. Basta atualizar o grafo, no fundo.

De seguida o processo “remove <id>”, envia um sinal SIGKILL ao PID do id, que está guardado no “servidor”, e assim o processo “node” é terminado.

Alteração de componentes

O “node”, sabe executar um comando, pois tem o nome dele guardado numa variável, e quando quer executar algo, executa o comando que está na variável.

Se o “controlador” receber um comando “change <id> newCmd”, este envia uma mensagem ao id, através do seu FIFO, e assim o “node” irá alterar a string que armazena o comando, alterando aquele que tinha quando foi criado o nó, pelo que foi passado por parâmetro <newCmd>.