Execução Agendada de Conjuntos de Programas

PROJETO DE SISTEMAS OPERATIVOS

Mestrado Integrado em Engenharia de Telecomunicações e Informática

Grupo 6:

- Carlos Miguel Ferreira Fonseca A81868
- Leandro Henrique Dantas Alves A82157
- José Eduardo da Silva Santos A82350

UNIVSERSIDADE DO MINHO | ANO LETIVO 2018/2019

Introdução

Neste projeto foi implementado um serviço de execução agendada de programas, em que o utilizador pode submeter, cancelar e listar agendamentos, definir um limite ao número máximo de agendamentos que poderão ser executados concorrentemente, e um endereço de correio eletrónico local para o qual deverão ser enviados os detalhes do resultado da execução dos agendamentos. O serviço é composto por um servidor e um cliente que comunicam através de *pipes* com nome.

Desenvolvimento

Agendamento

Para agendar uma execução de um programa, o utilizador (que controla o programa da agenda) escreve nos argumentos da mesma a opção "-a", a data e hora e a linha de comandos que pretende executar. A agenda converte a data e hora para formato de tempo Unix e junta-o juntamente com os restantes argumentos separados por espaços numa *string* enviada para o servidor através de um *fifo*.

No lado do servidor, este separa a *string* e adiciona cada elemento da tarefa (tempo e argumentos) a uma lista ligada de tarefas, enviando de seguida a posição dessa lista como identificador da tarefa à agenda.

Para agendar a execução da tarefa, o servidor calcula a diferença entre o tempo atual e o tempo agendado, se essa diferença for menor do que a diferença definida por tarefas anteriormente agendadas, então a nova tarefa será a próxima tarefa a ser executada, sendo definido um sinal de alarme com a nova diferença calculada e guardados os elementos dessa tarefa um *array*. Esse *array* é esvaziado e posicionado na posição 0 por ser uma diferença menor do que a anterior, mas se fosse igual (tarefas agendadas para a mesma hora, portanto concorrentes), a posição seria incrementada.

Listagem

Para o utilizador receber a listagem de todas as tarefas agendadas e executadas, envia para o servidor um único argumento "-l". O servidor percorre a lista ligada de todas as tarefas e separa-as em tarefas agendadas, se o tempo for futuro, e executadas, se for passado. Nesta listagem, em cada tarefa é mostrado o respetivo identificador, o tempo convertido de Unix para formato de data e hora e a linha de comandos.

Cancelamento

Para cancelar a execução de uma tarefa agendada, o utilizador envia para o servidor como argumentos "-c" e o identificador da tarefa. O servidor percorre a lista ligada das tarefas até chegar ao identificador que recebeu e substitui o primeiro caracter da *string* que contem o tempo por "*", fazendo com que a conversão dessa *string* para formato tempo seja com o valor 0, que é um valor ignorado na listagem, fazendo com que essa tarefa não seja mostrada na mesma.

Depois disso, a lista é percorrida novamente para se saber qual e quando é que a próxima tarefa deve ser executada (caso a que foi cancelada tenha sido a tarefa planeada a ser a próxima a ser executada). Os tempos passados são ignorados, incouindo os que tiverem valor 0, ignorando tarefas canceladas e executadas.

Execução

A execução de um programa acontece devido à interrupção do sistema devido ao alarme definido. Nessa interrupção, o programa entra num ciclo para executar todas as tarefas concorrentes, e em cada processo filho, o *standard output* e o *standard error* são redirecionados para um *pipe* anónimo correspondente, que faz parte de um *array* de *pipes* indexados pelo número de concorrência, fazendo com que sejam guardados o *standard output* e o *standard error* da execução da linha de comandos nos *pipes* que são lidos pelo processo pai e guardados em *arrays* de *outputs* e erros, também indexados pela concorrência.

Depois do ciclo das execuções, o programa entra num segundo ciclo para esperar que os processos morram e para passar o valor de saída dos mesmos para outro *array*, para adicionar os detalhes das execuções (identificador, data, linha de comando, valor de saída, *standard output* e *error*) a uma nova lista ligada de tarefas executadas e enviar os mesmos para o endereço eletrónico local definido.

Depois disso, a lista de todas as tarefas é percorrida para se saber qual e quando é que a próxima tarefa deve ser executada.

Consulta

Para consultar o resultado de uma tarefa já executada, o utilizador envia para o servidor os argumentos “-r” e o identificador da tarefa. O servidor percorre a lista das tarefas executadas até encontrar a que tem o identificador correspondente. Se o tiver, os detalhes da execução dessa tarefa (identificador, data, linha de comando, valor de saída, *standard output* e *error*) são enviados para o cliente.

Endereço eletrónico

Para definir o endereço eletrónico local que recebe os resultados de cada tarefa executada, o utilizador envia para o servidor os argumentos “-e” e o respetivo endereço. O servidor guarda esse endereço numa *string* e sempre que for executado um programa, os seus detalhes são enviados para o cliente cujo formato é a linha de comandos como título e os detalhes como conteúdo.

Num processo filho, o *standard output* é redirecionado para um *pipe* anónimo de escrita onde é escrito o *output* da execução do *echo* aos detalhes. No pai, o conteúdo do *pipe* é lido na execução do *mail* cujos argumentos são “-s”, a linha de comandos como título e o endereço definido. Esta ação requer a instalação do pacote mailutilis.

Limite de tarefas executadas concorrentemente

Para definir o limite de tarefas executadas concorrentemente, o utilizador envia para o servidor os argumentos "-n" e o número pretendido.

Sempre que for agendada uma nova tarefa, o servidor ao verificar se esta é concorrente com outra tarefa, este compara o número de concorrência com o limite definido se for ultrapassado, o agendamento é cancelado.

Além disso, sempre que a lista de tarefas for percorrida depois de uma execução ou cancelamento, se as próximas tarefas agendadas forem concorrentes, apenas as 'n' primeiras serão executadas e as restantes serão canceladas ('n' = limite definido).

Conclusão

Neste projeto, fomos capazes de aplicar e aprimorar todo o conhecimento acerca da matéria lecionada nas aulas da UC, fazendo com que algumas dúvidas da mesma fossem esclarecidas.

As nossas maiores dificuldades surgiram principalmente ao guardar os *outputs* e valores de saída na execução de tarefas concorrentes, mas foram ultrapassadas e achamos que concluímos este projeto com sucesso.