Sistemas Operativos

Trabalho Prático

Serviço de Agendamento de Tarefas

Grupo de Sistemas Distribuídos
Universidade do Minho

24 de Novembro de 2017

Informações gerais

- Cada grupo deverá ser constituído por até 3 elementos;
- O trabalho deverá ser entregue até às 23:59 de 11 de Dezembro de 2017;
- Deverá ser entregue o código fonte e um relatório de até 4 páginas (A4, 11pt) no formato PDF (excluindo eventuais capas e anexos).
- A apresentação do trabalho ocorrerá a 13 de Dezembro de 2017.
- O trabalho tem carácter obrigatório contribuindo no sentido de maximização do resultado obtido por frequência, representando nesses casos 30% da nota final.

Descrição do trabalho

Considere um serviço de agendamento de execução de tarefas que permita as seguintes operações:

- `agendar AAAAMMDD hhmm tarefa`: Agendar a execução de uma tarefa para uma data e hora. A tarefa corresponderá à execução de um comando, o qual será consituído por uma sequência de um ou mais programas encadeados por *pipes*. Cada programa poderá ter 0 ou mais argumentos. (p. ex: '`sort fich | uniq | wc -l`'). Assuma que não terá que tratar outros redireccionamentos oferecidos pela `bash`, como são o caso do '`>`', '`>>`', '`2>`', ou '`<`'. Esta operação deverá retornar um identicador único que ficará associado ao agendamento realizado. O identificador deverá ser um valor inteiro (valor inicial 0), incrementado a cada invocação desta operação.

- `listar agendados|cancelados|em-execucao|executados|todos`: Listar os agendamentos previamente realizados, consoante o estado especificado. A operação deverá retornar uma lista de tuplos constituidos pelos atributos 'identificador', 'date e hora', 'estado', e a 'tarefa' (comando) propriamente dita.

- `consultar identificador`: Consultar informação detalhade relativa a uma tarefa previamente agendada. Esta operação deverá retornar sempre os atributos referidos na descrição da operação '`listar`'. No entanto, no caso de tarefas já executadas, deverá ainda retornar o valor de saída e o conteúdo produzido no seu *standard output* durante a sua execução.

- `cancelar identificador`: Cancelar a execução da tarefa especificada.

Deverá ser também ser desenvolvido um servidor, mantendo em memória a informação relevante para suportar a funcionalidade acima descrita. Sempre que o servidor receber um sigal `SIGHUP` deverá cancelar todas as tarefas por executar. O servidor deverá considerar que uma tarefa terminou a sua execução apenas quando tiver terminado o último programa do respectivo encadeamento por *pipes*. Se necessitar de simplificar a implementação do seu servidor, considere então que uma tarefa é composta por não mais de 3 programas encadeados, e que cada programa não tem mais de 3 argumentos. O servidor deverá validar os argumentos de cada uma das operações, e deverá retornar os seguintes código de execuçãos: '0' no caso de sucesso, '1' no caso de 'argumento inválido', e '2' no caso de operação mal sucedida. Como valorização adicional do exercício, procure garantir que quando o servidor receber um sinal `SIGINT`, este cancelará todas as tarefas naquele momento em execução.

Deverá ser desenvolvido um cliente que ofereça uma interface textual com o utilizador e que permita suportar a funcionalidade acima descrita. Essa interface deverá ser por argumentos de linha de comando, ou, caso não sejam passados quaisquer argumentos, através de um interpretador de comandos semelhante ao oferecido pela `bash`.

O cliente deverá interagir com o servidor usando uma linguagem homogénea, orientada à linha de texto (protocolo).

Tanto o cliente como o servidor deverão ser escritos em C e comunicar via *pipes com nome*. Na realização deste projecto não deverão ser usadas funções da biblioteca de C para operações sobre ficheiros, salvo para impressão no *standard output*. Da mesma forma não se poderá recorrer à execução de comandos (directa ou indirectamente) através do interpretador de comandos (p. ex.: *bash*) ou da biblioteca standard de C (p. ex: `system()`).