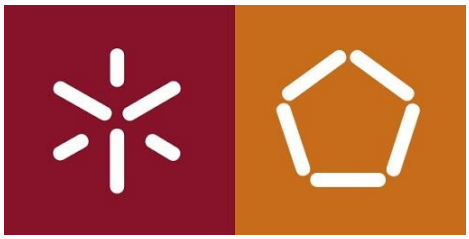

Escola de Engenharia

Departamento Informática

Mestrado Integrado em Engenharia Informática

Controlo e Monitorização de Processos e Comunicação

Sistemas Operativos

Grupo 114

Fernando Henrique Carvalho Lopes (A89472)

Pedro Almeida Fernandes (A89574)

Índice

Introdução

Este trabalho tinha como objetivo realizar um serviço de monitorização de execução e de comunicação entre processos. Para tal tarefa tivemos de colocar em prática o conhecimento adquirido ao longo deste semestre nesta unidade curricular. No nosso trabalho decidimos criar duas structs diferentes. A primeira, a estrutura args, que contém no seu interior um array de strings, serve para realizar parsing ao longo do trabalho, sendo que esta revela a sua maior utilidade e importância no que toca à realização de tarefas (opção -e p1 | p2 … | pn da linha de comando). A segunda struct, chamada de pidList tem um papel vital na nossa aplicação visto que é ela que guarda em memória toda a informação dos processos que vão sendo executados. Contém no seu interior uma variável para guardar o comando executado char *command, outra para guardar o pid do filho “master” (fillho criado para executar uma nova tarefa) int pid, uma outra para dizer o estado do processo (em execução ou terminado, sendo este modo diferenciada pela forma de término) int status , e por fim uma variável de controlo da estrutura int index.

Executar tarefa

Para a execução de uma tarefa existem duas maneiras, uma delas passando ao cliente o comando pelos argumentos (./argus -e “p1 | p2 … | pn”) ou uma outra quando quando não se passam argumentos a este (./argus), ele abre e podem se executar os comandos colocando de igual forma (-e p1 | p2 … | pn). Quando se vai a executar uma nova tarefa é realizado um fork() para a criação de um filho “master”. Este “master” cria por sua vez mais filhos para executar as diferentes tarefas introduzidas. Os filhos vão executando através de execvp o seu respectivo comando e de seguida passam o resultado ao filho seguinte através de pipes anônimos. Durante este processo a tarefa pode ser interrompida pelos tempo máximo num pipe, tempo máximo de execução ou mesmo pelo comando de terminar, sendo que em todos estes casos o filho “master” morre juntamente com os seus filhos. No final da realização da tarefa, o resultado é redirecionado do servidor para o cliente através de FIFOS.

Definir tempo máximo de execução

Nesta aplicação é permitido escolher um tempo máximo de execução para cada processo, sendo que este não é fixo e pode ser alterado quer pela linha de comando (./argus -m n), quer quando não se passa nenhum parâmetro a este (./argus) com a mesma opção (-m n), sendo que este n representa em segundos o novo tempo de execução. Esta variável altera o tempo do alarm() que é ligado após a criação do filho “master”. Se o tempo passar antes do processo executar o filho “master” e todos os seus filhos são mortos, e o status na estrutura pidLlst desse processo é alterado para 2, o código de indicação de término por tempo excessivo de execução.

Definir o tempo máximo de inatividade

Foi também especificado um tempo máximo de inatividade de comunicação entre pipes anônimos. É possível alterar esta variável usando o comando (-i n) quando feito (./argus) ou então também é possível passar por argumento na forma (./argus -m n), sendo que o n corresponde ao máximo de segundos de inatividade que o utilizador pretenda. Para a realização desta tarefa foi colocado um alarme em cada filho do filho “master”, caso algum ativar a tarefa é interrompida e o filho master obtém essa informação na fase de espera dos filhos. Nessa fase ele verifica como cada um dos seus filhos terminou, e se encontrar algum altera a variável status na estrutura para 1, o valor corresponde à terminação do processo por tempo excessivo no pipe anónimo.

Terminar tarefa

É nos permitido terminar uma tarefa de forma manual, daí a função desta opção, que pode ser obtida fazendo (./argus -t n) ou quando não são passados argumentos a este com o comando (-t n) sendo que este n representa o número do processo que se pretende terminar. Após esta indicação é enviado um sinal SIGUSR1 ao filho “master” correspondente à tarefa n, que vai resultar no seu suicídio e na morte dos seus filhos. É então após a sua morte alterada a informação status da estrutura pidList do processo n para 3, o código utilizado por nós para referir as tarefas “assassinadas” pelo comando -t.

Apresentar ajuda

Todas as aplicações devem demonstrar ao utilizador como devem ser utilizadas e é por isto que existe esta opção. Fazendo (./argus -h) ou começando argus sem argumentos e apenas escrevendo (-h) é impresso ao cliente a seguinte lista de instruções :

-i n -> tempo inatividade segs
-m n -> tempo execucao segs
-e p1 | p2 ...| pn -> executar o comando p1 | p2 ... | pn
-l -> listar as tarefas em execução
-t n -> terminar processo n
-r -> historico de processos terminados
-h -> ajuda

Esta informação é útil uma vez que torna mais fácil para um utilizador se adaptar de forma rápida ao modo de utilização da aplicação e também ajuda caso este não saiba a forma como utilizar um ou vários dos comandos que são dispostos, isto pode variar entre não saber os argumentos que cada um recebe até a não saber como chamar o comando desejado.

Listar registo histórico

Tal como listar as tarefas em execução, listar todas as tarefas terminadas também é algo que se espera de um sistema de controlo de processos. Assim sendo existe o comando (-r) ou caso se passe como argumento (./argus -r) que realiza isso mesmo. Mostra ao cliente todos as tarefas já terminadas especificando o modo como terminaram na forma (#2, concluida: ls -ltr) para tarefas que concluíram a sua execução, (#2, max inatividade: ls -ltr) para tarefas que passaram o tempo limite num pipe, (#2, max execução: ls -ltr) para tarefas que ultrapassaram o tempo limite de execução e por fim (#2, terminada: ls -ltr) quando as tarefas foram concluídas por ação de -t. Tal como na lista anterior esta informação é obtida a partir da estrutura pidList, usando a variável status para saber em que estado terminou, 0 execução normal, 1 tempo máximo de inatividade num pipe atingido, 2 tempo máximo de execução ultrapassado e por fim 3, processo terminado com o comando -t.

Listar tarefas em execução

Ver quais os processos em execução é de facto algo muito importante para a monitorização de processos daí existir esta opção. A opção pode ser obtida utilizando o comando (-l) quando é executado argus sem argumentos ou pode ser passado como argumento (./argus -l). Esta opção mostra todos os processos em execução na forma (#1: cut -f7 ‘d: /etc/passwd | uniq | wc -l). Para tal percorre a estrutura pidLIst e para todos os processos que se encontrarem com status igual a -1 (código do status para processos em andamento) retira a informação necessária e imprime no cliente a partir de FIFOS.

Verificação de resultados

Durante a realização do trabalho tentamos sempre testar cada uma das funcionalidades, quer através de comandos fornecidos no enunciado, quer por uma script que fizemos para teste.

Para a funcionalidade de ajuda fomos simplesmente fazendo o comando (-h) quando iniciado o cliente por (./argus) ou (./argus -h) e verificando se o servidor nos envia a lista de ajuda que tínhamos realizado.

Para ambas as opções do tempo verificámos que a mudança do tempo estava a ser realizada, uma vez que quando as alteramos, o tempo com que a tarefa era morta era diferente.

Em relação à funcionalidade de terminar uma tarefa, primeiramente corríamos a nossa script, de seguida era corrido (-t) a essa tarefa e era verificado que a sua variável status mudava para 3, o número que definimos para representar tarefas mortas por (-t).

Tanto na opção (-l) e (-r) eram executadas várias tarefas que cujo o tempo de realização fosse relativamente elevado, como por exemplo a nossa script, e verificamos se o estado real do processo era igual ao esperado.

Finalmente na verificação dos resultados da opção (-e) foram corridos diversos códigos diferentes, quer na linha de comandos quer na aplicação desenvolvida de modo a verificar se os nossos resultados eram os mesmos. Foram usados entre outros os seguintes comandos :

- cut -f7 -d: /etc/passwd | uniq | wc -l
- ./exe (a nossa script)
- ls -ltr
- grep -v ^# /etc/passwd | cut -f7 -d: | uniq | wc -l
- echo batata
- sleep 10
- clear

Conclusão

Com a realização deste trabalho foi-nos permitido cimentar e conectar todos os conteúdos práticos lecionados neste semestre. Tivemos a oportunidade de testar numa forma prática o uso de funções que nos anos anteriores não seríamos capazes de o fazer. Funcionalidades como a criação de um servidor (argusd.c) que continuamente receberia comandos até este não ser mais necessário, aproximado a um *deamon*, um conceito que anteriormente seria totalmente desconhecido. Para além disto o uso de pipes para comunicar entre processos e conseguir realizar vários destes ao mesmo tempo, revelou-se como uma nova forma, simples mas eficiente de realizar estas tarefas.

Em geral realizar este trabalho foi uma forma de melhorar e experimentar coisas novas, já que nos permitiu aprender a corrigir erros à medida que eles iam aparecendo.