U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Relatório Mini Projeto 1

Ferramenta XMOD

Trabalho realizado no âmbito da Unidade Curricular de **Sistemas Operativos**

André Pereira up201905650
Francisco Rente up201906761
Margarida Vieira up201907907
Tomás Vicente up201904609

Turma 2 Grupo 4

INTRODUÇÃO

O código que implementa a ferramenta ***xmod***, que tem como referência o comando ***chmod*** (*change file mode bits*) que permite modificar as permissões de acesso a ficheiros e diretórios, foi compartimentalizado em vários módulos funcionais que serão explicados, com algum detalhe abaixo.

TIME_CTRL

O módulo ***time_ctrl*** implementa as funcionalidades relacionadas com o tempo de execução do programa, nomeadamente a obtenção do tempo em que o processo do topo da hierarquia iniciou a sua execução – ***getStartTime*** – e a obtenção do tempo decorrido desde o mesmo até à data da invocação da função – ***getMillisecondsElapsed***.

A primeira acede à informação do ficheiro criado para o processo do topo da hierarquia – *"proc/[pid]/stat"* – que diz respeito ao último acesso do processo ao ficheiro. A informação é guardada numa variável do tipo *timespec* que oferece a vantagem de uma precisão de nanossegundos. A segunda, tal como o nome indica, lê o tempo atual e retorna, em milissegundos, o tempo decorrido.

LOG

O módulo ***log*** trata de verificar se o *path* para o *logfile*, onde é suposto ir sendo registado, ao longo da execução do programa e mediante a ocorrência de certos eventos específicos, um conjunto de informações relativas aos mesmos, se encontra definido na variável de ambiente *LOG_FILENAME* – ***openLogFile***.

No caso do *path* se encontrar definido, o ficheiro será aberto ou criado, se ainda não existir. A função recebe como parâmetro uma *flag booleana* – *truncate* – que determina o tipo de abertura do ficheiro, garantindo, desta forma, que o mesmo só é truncado se o processo que o estiver a abrir for o do topo da hierarquia. Qualquer processo gerado a partir deste, apenas acrescentará informação à já contida no ficheiro.

As funções ***log****, ao invocarem a função ***logEvent***, tratam de registar a ocorrência de um determinado evento do conjunto especificado no enunciado. Esta última recebe o tipo de evento – variável do tipo *event_t* – e uma *string* com informação adicional que, dado que depende do tipo de evento a ser *logged*, é preparada na função específica do mesmo, *i.e.* ***logProcessCreation***, ***logSignalReceived***, etc.

Por último, a função ***closeLogFile*** trata de, antes da terminação de qualquer processo, fechar o *logfile*.

PARSE

O módulo ***parse*** tal como o próprio nome indica, implementa a funcionalidade de *parsing* do comando introduzido na linha de comandos, através da função ***parseCommand***.

Toda a informação relativa ao comando é processada e guardada numa variável do tipo *command_t*. Esta função tem em conta a possibilidade de ocorrência de erros de input e, nesses casos, é apresentada no ecrã uma mensagem a alertar o utilizador e a função retorna um valor indicativo da ocorrência de erro. Em consequência, o próprio programa é abortado na função *main*.

IO

O módulo ***io*** contempla as funções responsáveis por enviar para a *standard output* as mensagens resultantes da execução da ferramenta xmod, considerando os vários casos de uso possíveis.

Contempla, para além dessas, a função ***printCurrentStatus***, que é invocada quando o utilizador premie CTRL-C (ou, por outras palavras, envia o sinal SIGINT ao programa) e é responsável por mostrar no ecrã o estado atual de execução do programa, *i.e.* o número de ficheiros encontrados até ao momento, o número de ficheiros já modificados, etc.

UTILS

O módulo ***utils*** agrega um conjunto de métodos utilitários que implementam funcionalidades usadas frequentemente ao longo do restante código.

A mero título de exemplo, a função ***isParentProcess*** verifica se um processo é ou não o do topo da hierarquia – comparando o identificador do processo *caller* com o identificador do seu grupo de processos; a função ***convertIntegerToString*** converte um número inteiro, recebido como parâmetro, para formato *string*, retornando o resultado na variável *dest*, igualmente recebida como parâmetro; a função ***leave*** recebe um valor inteiro correspondente àquele que será o valor de retorno

da função que a invoca e é responsável por esperar pela terminação de todos os processos filhos e, só posteriormente, dar ***exit*** ao programa com esse mesmo valor de terminação.

SIGNALS

O módulo ***signals*** incorpora funções relativas ao tratamento de sinais e à adoção do comportamento mais adequado aquando da sua receção. Para que exista o mínimo de *overhead* possível na resposta a uma dada sinalização, foram utlizadas funções *async-signal-safe* nos próprios *signal handlers* (*p.e. write*), para que estes possam atender aos pedidos e realizar os registos necessários de forma imediata.

Foram contabilizados todos sinais POSIX, sendo os de maior relevância:

- SIGINT: no caso do processo pai, questiona o utilizador para que este decida se quer ou não prosseguir com a execução do programa;
- SIGSTOP: enviado para pausar uma instância de um processo filho;
- SIGCONT: prossegue com o funcionamento de um processo filho, após este ter recebido o SIGSTOP;
- SIGTUSR1: ocorre quando o utilizador decide terminar o programa, sendo enviado pelo pai a todos os outros processos do mesmo grupo;

Para evitar a criação de múltiplos *handlers*, o ***genericSignalHandler*** é atribuído a todos os sinais que podem ser recebidos, excetuando os que são, de alguma forma, impeditivos ao procedimento das chamadas (SIGCHLD e SIGUSR1). Nesta função é verificado qual foi o sinal recebido e se o processo em causa é o pai, o que vai determinar o comportamento esperado a cada situação (*p.e.* no caso de SIGINT e processo pai, chamar a função ***parentSigintHandler***). De forma a ser possível realizar a anotação do sinal recebido, é fundamental, tanto no *handler* genérico, como nos *handlers* específicos, realizar o *log* e, posteriormente, efetuar a chamada ***raise*** para que o resultado final seja o procedimento *default* de um dado sinal.

De modo a que a hierarquia de processos se mantenha, estão colocados *loops* de espera (recorrendo às *system calls* ***wait*** e ***waitpid***) e são contabilizados os processos filhos, para que as pausas e os términos se adequem à sua ordem inicial de criação, evitando que se gerem processos órfãos e a desorganização da escrita no terminal e no ficheiro.

XMOD

O módulo ***xmod*** merece especial destaque, no sentido em que é responsável por agregar todas as peças que já foram sendo mencionadas acima.

A principal função deste módulo é a ***changeMode***. No caso de não ter sido especificada a flag que ativa o modo recursivo, '-R', apenas as permissões do ficheiro/diretório correspondente ao path introduzido pelo utilizador são alteradas – ***changeFileMode***. Caso contrário, todo o diretório (excetuando *symbolic links*), se se tratar efetivamente de um, é processado – ***changeFolderMode*** e, por cada subdiretório encontrado, é criado um novo processo que fica encarregue pelo processamento deste. Após a criação de cada subprocesso, a imagem desse é substituída por uma nova através da chamada à função ***execv*** que recebe os mesmos argumentos da linha de comandos, à exceção do *path*, que é atualizado. Em qualquer um dos cenários, é efetuada uma chamada ao ***chmod*** para alterar efetivamente as permissões do ficheiro/diretório em causa.

CONTRIBUIÇÕES

André Pereira	25%
Francisco Rente	25%
Margarida Vieira	25%
Tomás Vicente	25%