U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Sistemas

1º Mini Projeto

Ferramenta para modificar permissões de ficheiros

Beatriz Aguiar
Bruno Gomes
Matilde Oliveira
Rodrigo Tuna

Descrição

A ferramenta **xmod** é uma versão simplificada do comando **chmod** (*change file mode bits*), que permite modificar permissões de acesso a ficheiros e diretórios.

Estrutura do código

O projeto está dividido em três *source* files: xmod.c, auxMode.c e logFile.c.

logFile

Módulo responsável por inicializar e efetuar os registos dos eventos num log file[1].

A função **initLog** é chamada uma única vez, no primeiro processo, e consiste na criação de uma variável de ambiente **xmodStartTime** que irá servir como meio de transmissão do instante de tempo inicial do processo. No caso de já existir um ficheiro **LOG_FILENAME**, este é truncado.

Cabe à função **writeLog** adicionar o registo dos eventos que ocorrem no programa, no formato especificado pelo enunciado. Por uma questão de simplicidade, optamos por guardar os eventos num *enum* **logEvent**.

xmodAux

Este módulo reúne as funções auxiliares ao módulo principal.

A função **getFlags**, tal como o nome indica, retorna uma *mask* das *flags* passadas como argumento, pelo utilizador. A cada *flag* está associado um *bit*, e *mask* retornada será contruída pela reunião dos *bits* de todas as *flags* argumento.

A função **getSymbolic** constrói a *string* correspondente ao modo passado como argumento (ex.: modo octal: 0777; *string*: "rwxrwxrwx").

Obtemos o tempo de execução (tempo do processo atual - tempo do 1º processo) em *ms* na função **timeDifferenceMS**.

Como auxiliar à função **writeLog**, a função **getArgStr** concatena os argumentos recebidos na main, numa *string* que é posteriormente adicionada ao *log file*.

[1] Caso a variável **LOG_FILENAME** não exista, as funções deste ficheiro não serão chamadas.

xmod

Este é o módulo principal do programa. Com base nos argumentos recebidos, são construídas as variáveis *flags* e *mode*, necessárias à chamada ao sistema **chmod**.

No início são definidos os *handlers* na chamada à função **receiveSignals** e inicializadas as variáveis **processData**, **hasLog.** A primeira é uma *struct* que contém os dados **nMod**, **nTotal** e **starTime**, **currentDirectory** relativos a cada processo e importantes para a impressão no *logFile* ou no *stdout*, quando necessário[2]. **hasLog** é um boleano que verifica a existência, ou não, da variável de ambiente **LOG_FILENAME**.

De acordo com o método de invocação de xmod (octal ou "simbólico"), o programa irá ter dois rumos possíveis. No primeiro caso, octal, a *string* passada como argumento é convertida em octal (variável *mode_t*) e é efetuada a chamada ao sistema.

Quanto ao segundo caso, antes de chamar **chmod**, irá ser chamada a função **symbolicXmod** que irá converter (através de operações binárias) uma *string* do tipo "a+rwx" num número em octal.

Caso o número de argumentos seja superior a dois, isto é, caso sejam enviados argumentos suficientes, o programa prossegue. As flags, se existirem, são obtidas na chamada da função **getFlags** e o programa age consoante o pedido.

No caso de -v e -c, a função **xmod** imprime para o *stdout* a informação especificada por cada *flag.* Quanto à *flag* -R, se esta estiver ativa e o ficheiro passado como argumento for um diretório, irá ser chamada a função **goThroughDirectory** que percorrerá todos os ficheiros e, na eventualidade de encontrar um novo diretório, será criado um novo processo que irá executar novamente (através do comando **execvp**) o **xmod**, neste novo diretório. O processo pai irá continuar a percorrer, se existirem, os restantes ficheiros do diretório em que se encontra.

Se durante o decorrer do programa for mandado algum sinal, este irá ser enviado para o seu *handler* e se a variável **hasLog** estiver a 1, este evento será registado no **logFile**. No caso do **SIGINT**, os processos filhos são pausados (mandam a si próprios o sinal **SIGSTOP**) e o pai ficará responsável pela interação com o utilizador ("*Do you want to continue? (y or n)*"). No caso da resposta ser afirmativa, é enviado um sinal (**SIGCONT**) ao grupo que permitirá a continuação do programa, caso contrário o programa termina, com o envio do sinal **SIGUSR1**. O *handler* deste sinal tratará dos outputs para o logFile e terminará o processo com o sinal **SIGKILL**.

Num término normal do programa, todos os pais esperam pelos seus filhos.

[2] Se as flags -c ou -v forem passadas como argumento, ou se **LOG_FILENAME** estiver definida.

Participação

Beatriz Aguiar	up201906230	25%
Bruno Gomes	up201906401	25%
Matilde Oliveira	up201906954	25%
Rodrigo Tuna	up201904967	25%