U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

XMOD

Mini-Projeto 1

Sistemas Operativos 2020/2021

Turma 7 Grupo 1

Adriano Soares up201904873@fe.up.pt
Diogo Maia up201904974@fe.up.pt
Francisco Cerqueira up201905337@fe.up.pt
Pedro Pereira up201905508@fe.up.pt
Vasco Alves up201808031@fe.up.pt

Índice

Contexto

Requisitos Funcionais

O xmod visa ser um comando, tal como o chmod, capaz de mudar a permissão de um ficheiro / diretório ou, dependendo das opções, mudar as permissões de um diretório e do seu conteúdo. Este recebe, pela seguinte ordem, uma ou mais opções de funcionamento, um novo modo de permissões, sendo este em octal ou regular, e o ficheiro / diretório que terá as suas permissões modificadas.

O formato requerido será então:

xmod [OPÇÕES] <MODO> <FILE/DIR>

1) OPÇÕES (zero ou mais):
    a) **-v :** modo verboso, que escreve todas as operações de modificação de permissões quer tenham ou não sido alteradas;
    b) **-c :** modo que escreve apenas as operações de modificação de permissões que sofreram alteração;
    c) **-R :** modo que permite que o conteúdo do diretório passado como parâmetro também sofra as alterações especificadas nas suas permissões.
2) MODO (apenas um):
    a) Octal: É representado no formato "0xxx" sendo que o zero inicial é obrigatório e o 'x' pode tomar qualquer valor de 0 a 7. Este conjunto de algarismos representa as permissões que serão embutidas no ficheiro especificado. Para mais informação sobre este tópico, aceda: https://linuxize.com/post/chmod-command-in-linux/ que estende a informação acerca do octal.
    b) Regular: modo que especifica uma string com permissões que pretende alterar e que operações quer executar sobre elas. Por exemplo "u+rw" permite adicionar ao detentor do ficheiro a permissão de escrever e ler. Para mais informações sobre a sua formatação e explicação detalhada, aceda: https://linuxize.com/post/chmod-command-in-linux/.
3) FILE/DIR : caminho relativo ou absoluto de um ficheiro ou diretório.

Funcionalidades Adicionais

Como requerimentos extra, o xmod deve registar num ficheiro especificado na variável ambiente LOG_FILENAME todos os eventos que forem desempenhados pelo programa como a modificação das permissões, o levantamento de um sinal e o final de um processo. Este requerimento não é obrigatório.

O xmod também trata os sinais que recebe e, ao receber o sinal SIGINT, ao contrário do chmod, este pausa todos os processos menos o pai e regista na consola quantos ficheiros o processo já encontrou e modificou. Depois de todos os processos o fazerem, o utilizador pode escolher se quer terminar o programa ou continuar o seu trabalho.

Por fim, o xmod utiliza, no modo de recursão, processos-filho de modo a mudar as permissões dos subdiretórios encontrados ao longo do diretório principal. É importante referenciar este aspeto, pois, neste modo, o processo pai é obrigado a esperar que todos os seus filhos terminem, não deixando processos-zombie pelo caminho.

Estrutura Geral do Código

Atendendo às necessidades básicas do xmod, nós desenvolvemos código capaz de interpretar e avaliar os parâmetros apresentados pelo utilizador na execução do comando. Este avalia tanto a sua posição, como a sua autenticidade, comportando-se adequadamente caso as opções de funcionamento, o novo modo de permissões e / ou o FILE/DIR não estiverem dentro dos limites estipulados pelo comando.

Este código está dividido em cinco grandes módulos. Estes são: o xmod.h, responsável pelo método *main()* do xmod; o xmod_info.h, que interpreta e trata os parâmetros apresentados pelo utilizador, guardando-os de modo a permitir a sua fácil e rápida manipulação; o change.h, módulo responsável por alterar as permissões do(s) ficheiro(s) em causa; o regfile.h, que permite o registo de qualquer evento, desde a modificação de um ficheiro até à receção de um sinal por parte de um processo, e, por fim, o signal.h, responsável pelo tratamento dos sinais que são levantados ao longo da execução do programa.

Módulos Utilizados

xmod_info.h

Este módulo, como dito anteriormente, permite a leitura e tratamento dos dados fornecidos ao comando. Para tal efeito, foi criada uma estrutura *XmodInfo* que guarda informações relevantes, como o nome do ficheiro, o modo pré-mudança, o modo-pós-mudança e as opções especificadas, que, por sua vez, são guardadas numa nova estrutura *XmodFlags*.

Para além da função principal deste módulo que permite o preenchimento desta *struct*, também temos várias funções pequenas que facilitam o trabalho desta. Estes métodos desempenham papéis importantes, como converter o modo em octal (*mode_t*) e identificar em qual modo está o parâmetro que foi disponibilizado.

signal.h

O signal.h é o módulo responsável por tratar da maioria dos sinais e permitir que, no momento em que um processo os receba, os processe. A maioria das funções do signal.h são *handlers* de sinais, como o SIGINT e o SIGUSR1.

Nós decidimos destacar estes por serem os que desempenham os papéis mais importantes no tratamento de sinais.

O SIGINT permite a interrupção, temporária ou permanentemente de todos os processos e o registo na consola do desenvolvimento de cada um na mudança das permissões dos ficheiros a seu cargo.

O SIGUSR1 permite que, se o utilizador preferir terminar o programa precocemente após a chamada do SIGINT, cada processo notifique aos seus processos-filhos a necessidade de se terminar, matando todos os processos-filho.

Alguns sinais com "handler" definido são o SIGCHLD e o SIGCONT.

regfile.h

O regfile.h está encarregado de monitorar a LOG_FILE. Este módulo tem os métodos que permitem inicializar, escrever e terminar o ficheiro nela indicado.

Neste módulo existe uma função para cada evento como a modificação de um ficheiro e a receção de um sinal.

change.h

O módulo change.h está encarregado de alterar as permissões do ficheiro/diretor. Este módulo é principalmente responsável pela mudança de permissões, incluindo o modo recursivo, que atribui um novo processo sempre que necessário e, no final, aguarda por todos os filhos.

Uma função que desempenha um papel importante no xmod é *initProcess()*, que determina se o processo atual é o primeiro processo criado ou um dos processos-filho e esta distinção é alcançada através de uma variável ambiente criada pelo próprio programa.

Chegamos a este método após ponderar sobre várias metodologias possíveis, tais como:

A. A utilização de uma variável global no processo-filho antes de chamar a função *execv()*, que revelou não ser possível, pois o processo perdia todos os dados após a chamada da função;
B. O uso de sinais, que julgamos ser bastante imprevisível, pois nunca teríamos a garantia de que o pai iria correr primeiro que o filho e o filho poderia não receber o sinal;
C. A utilização de *pipes*, que permite a sincronização dos processos.
   No entanto, não seriam simples de implementar dada a estrutura do nosso trabalho e o facto de esse módulo ter sido dado quase no final da data de entrega do projeto;
D. O uso de um ficheiro auxiliar, que garantiria a sincronização da informação entre processos, mas que criaria um efeito "pop-up" para o utilizador, não sendo por isso utilizada.

Não podemos deixar de sublinhar o facto de que, a partir dos métodos deste módulo, são manipuladas cinco variáveis globais, que, em nenhum momento, são alteradas fora deste módulo, mas são necessárias para o registo de eventos e tratamento de sinais.

xmod.h

Este módulo, que contém apenas o método *main()* está encarregado de recorrer aos outros módulos e juntar todas as partes do projeto.

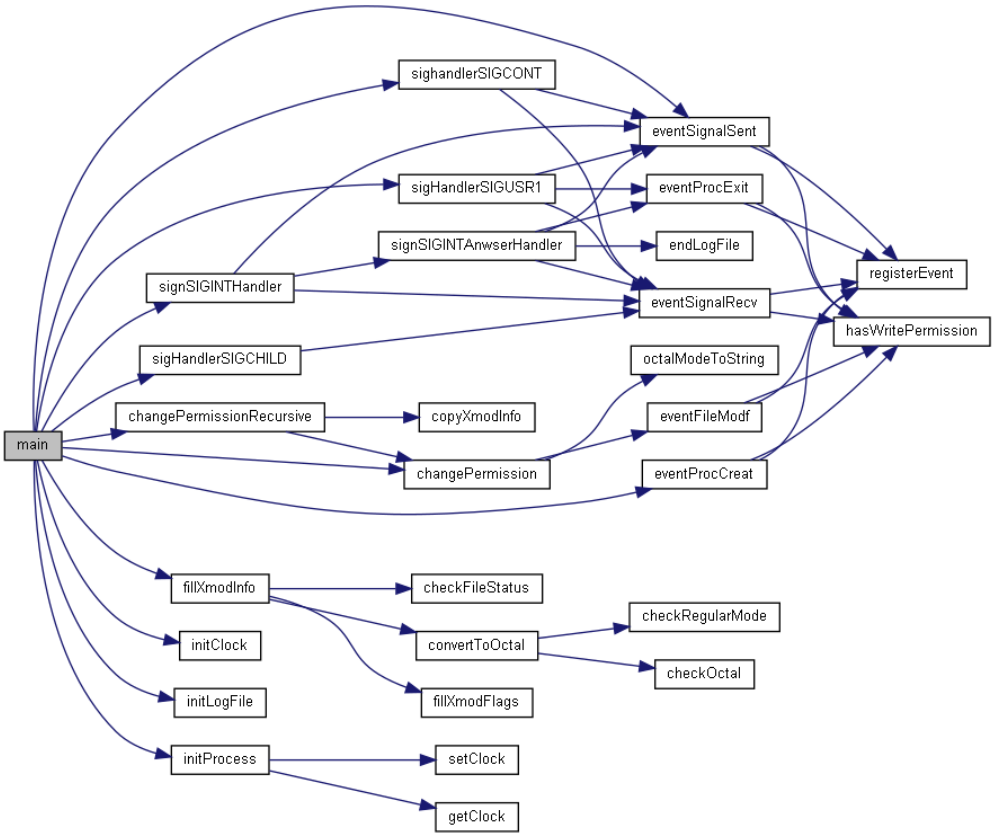

Figura 1. Gráfico de chamadas de funções da função *main()*

Contribuição Percentual

Nome	Percentagem
Adriano	20%
Diogo	22%
Francisco	22%
Pedro	18%
Vasco	18%