

Faculdade de Engenharia da Universidade do Porto
MIEIC - SOPE
2020/2021

Relatório Mini-Projeto 1

xmod

Turma 6 - Grupo 3
Adelaide Santos
Bruno Mendes
Eunice Amorim
Rita Mendes

up201907487
up201906166
up201904920
up201907877

Processamento de Input:

Para o correto funcionamento de *xmod*, é necessário prévio processamento de input, semelhante a *chmod*. Desta forma, não se incorre no risco de executar kernel calls com parâmetros desmesurados. Posto isto, são permitidas algumas sintaxes avançadas semelhantes às suportadas pelo utilitário *chmod*, não requeridas pelo enunciado:

1. Modos simbólicos podem combinar-se, desde que separados por vírgulas, sem espaços (*xmod a+x,u-r <file>*);
2. *Flags* de opções podem ser introduzidas separadamente, desde que antes do path do ficheiro (*xmod -R -c 0777 <file>*);
3. O tipo de utilizador pode ser omitido, ficando 'a' implícito (*xmod +x <file>*).

Ao contrário, porém, de *chmod*, multiple file paths não são permitidos, nem modos octais podem perder o zero inicial, como sugerido no enunciado. Outros erros de sintaxe são assinalados como em *chmod*.

Leitura do modo de permissões do ficheiro:

De maneira a que seja possível a utilização dos símbolos '+', '-' e '=', para calcular o modo desejado, é necessário ler o modo de permissões dos ficheiros antes de efetuar a chamada de sistema *chmod()*. Assim, foi implementada uma estrutura que guarda e trata essa informação, permitindo ao resto do programa aceder ao *path* relativo, modo de permissões e ao tipo de ficheiro.

Sinais:

Como não foi encontrado um modo *standard* de encontrar todos os sinais possíveis, foi definido um handler para todos os sinais com número entre 1 e 32 que estejam definidos no sistema. Tal decisão foi tomada visto que, na maioria dos sistemas, estes correspondem aos sinais mais comuns.

Outra questão relevante é a associação entre sinais e logs. Se num log o pid associado ao evento for 0, significa que esse sinal está a afetar todo o grupo de processos. Por outro lado, o nome apresentado dos sinais nos logs não corresponderá exatamente ao nome dos macros, devido uso da função *strsignal()*, por conselho do Professor José Magalhães Cruz, evitando definir nomes *hard coded*.

Logs:

A utilização de logs foi fulcral para o desenvolvimento e debug do projeto. Por ser tão frequente e importante, incluiu-se uma função *logger* para cada evento, que chama um *logger* genérico interno com os parâmetros adequados. Para facilitar a passagem de argumentos, apostou-se numa estrutura *union* que guardará apenas os argumentos necessários para cada chamada.

Outras melhorias foram incluídas. Usando *lseek()*, para colocar o apontador no fim do ficheiro de cada vez que for necessário concatenar um novo registo, conseguiu-se utilizar um único *open()* e um único *close()* para cada processo. Recorreu-se também a um *file lock* para evitar a escrita concorrente do ficheiro de registo por parte dos diferentes processos.

Modo verboso:

Comportamento semelhante a *chmod*, em que se incluem alguns pormenores:

1. Em caso de falha à chamada *chmod()*, apresenta-se “failed to change mode from <old> (...) to <new> (...)”, ficando old e new com valores indefinidos;
2. Mudando as permissões de uma diretoria para tais que impeçam a modificação de permissões dos seus ficheiros, é apresentado algo semelhante a: “changed mode successfully <dir>”; “cannot read <dir>”; “failed to change mode of <dir>”, o que é realmente peculiar, mas foi exatamente replicado.

Participação de cada elemento:

O esforço de trabalho foi dividido por todos de igual forma.

#teamWork