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

Relatório Mini-Projeto 1
xmod

Adelaide Santos up201907487
Bruno Mendes up201906166
Eunice Amorim up201904920
Rita Mendes up201907877

Processamento de Input:

Para o correto funcionamento de *xmod*, é necessário prévio processamento de input, semelhante a chmod, para não incorrer no risco de executar kernel calls com parâmetros desmesurados. Posto isto, permitimos 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:

Para 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 usados.

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 ao nome dos macros, por uso da função *strsignal()* por conselho do Professor José Magalhães Cruz, para não definir nomes “hard coded”.

Logs:

A utilização de um registo de log foi fulcral para o desenvolvimento e debug do projeto. Daí optou-se por incluir abstrações que facilitam as chamadas: uma função logger para cada evento que chama um logger genérico interno com os parâmetros adequados. Para facilitar a passagem de argumento, 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 cada vez que foi necessário concatenar um novo registo, conseguiu-se usar um único *open()* e um único *close()* para cada processo. Adicionou-se um lock para evitar a escrita concorrente.

Modo verboso:

Comportamento semelhante ao 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 do tipo: “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