Índice

Sobre o *Knowledge Tester*

Este é um *software* que tanto pode ser usado para auto-estudo, como para ensino/formação, seja presencial e remoto, e até para execução de exames remotos protegidos por encriptação *Rijndael 256 bits (AES 256)*, que desenvolvi nas minhas linguagens de programação favoritas (*C/C++*), para o meu sistema operativo favorito (*Linux*).

Talvez um dia mais tarde com tempo, crie uma interface gráfica para o mesmo, e o exporte para *Windows* também (apenas a parte gráfica). A versão de consola será apenas para *Linux*.

A ideia de o desenvolver teve origem na vontade que tive em 2019 em tirar mais de 20 certificações informáticas, e necessitar de uma ferramenta para assimilar mais rápido as milhares de páginas necessárias para as concluir entre um a dois anos, numa altura em que estava com ideias de dedicar cinco anos da minha vida apenas à Ciber-Segurança, como forma de despedida, antes de me dedicar de vez à carreira na área da Formação.

Não o tinha começado a desenvolver porque nesse ano decidi dedicar-me logo à Formação, por já ter tido várias experiências na área de *Cyber-Security*, e achar que não eram necessários mais cinco anos, além de que sempre quis dedicar-me um dia à Formação.

Durante um curso iniciado no ano de 2019, que era necessário frequentar para os planos de ser formador, dado que já conhecia bem a matéria que estava a dar e tinha tempo livre, dediquei o tempo à Programação, a criar coisas como o meu *game engine*, entre outras coisas.

Nesse curso, tive uma colega invisual, que tinha dificuldades em aprender certas matérias, mas que tinha mais facilidade em usar o *Linux*, devido às muitas ferramentas que tinha, em especial *text-to-speech* em *shells*, e para a ajudar também, comecei a desenvolver o *software*, que teve como origem a folha abaixo de rascunhos escrita numa dessas aulas.

Como se pode ver pela imagem abaixo, o sistema foi imaginado de forma muito simples, baseado em tags, conforme explicado abaixo na secção sobre criação de exames.

Podem reparar nas tags “[Q]” (de “*Question*”), “[I]” (de “*Info*”), as “[1]” e “[0]” para perguntas certas e erradas, etc.

Neste tutorial explicarei como usar o *software* para auto-estudo, para criação de exames, para treinar para certificações, como encriptar exames, criar exercícios, etc.

Este é um manual inicial, com o tempo à medida que o programa evolua, ele será também melhorado.

O uso do *Knowledge Tester* para auto-estudo

Este *software* pode ser usado para auto-estudo em vários níveis, e é especialmente útil para quem estuda para certificações *IT*, que como todos sabemos, se baseiam muito em "testes de cruzinhas", mas permite também respostas directas multi-linha para quem queira praticar para certificações práticas como por exemplo a *Red Hat Certified Engineer*, entre outras.

Ela pode ser usada também noutras áreas e disciplinas, pois permite tanto com perguntas directas como cruzes, entre outras funcionalidades a desenvolver no Futuro, o criarmos os nossos próprios exames.

O *K.T.* como ferramenta de aprendizagem acelerada

Este *software* permite aos formandos/alunos aprender de forma acelerada, através de vários métodos e ajudas, como o enviar perguntas falhadas para o fim da fila, o "modo mental", entre outras funcionalidades, ou até pelo facto de uma pessoa estar a aprender enquanto joga, que é uma forma natural de aprender, e que leva a que alunos possam estar a competir entre eles para ver quem tem melhores notas, e sem se aperceberem memorizando comandos.

Permite assim com 10 minutos de treino obter a prática que necessitaria de várias horas de uso real dos sistemas em causa para aprender.

O *Knowledge Tester* para ensino e exames presenciais

Com este *software*, pode-se formar/ensinar também, pois podemos incluir ajudas em cada questão, para que o formando possa aprender sobre o porquê de ter falhado uma resposta, e até praticar depois num terminal ao lado, podendo sempre tirar dúvidas com o formador no local.

E dado que permite a criação de exames encriptados, que geram relatórios encriptados e que não poderão ser alterados, e que terão de ser devolvidos ao formador no fim do exame, pode ser usado para a execução de exames com avaliação imediata, sem a possibilidade de falhas de segurança nos mesmos.

A avaliação automática dos exames ajuda na tarefa também, e será melhorada posteriormente.

O *Knowledge Tester* para ensino e exames remotos

Da mesma maneira que no ensino/formação presencial, dado que os exames são encriptados com *AES 256*, que impede aos alunos a alteração do relatório gerado pelo exame, podem os relatórios ser devolvidos ao formador ou professor após o término do exame, sem o risco de serem alterados.

É assim útil no ensino e formação à distância, e até exames à distância, com total segurança.

Instalação

Este *software* não requer qualquer instalação, basta que o utilizador faça *download* do mesmo, algures na página do autor (*www.goncalo.pt*), e mova o ficheiro para uma pasta à escolha, como neste caso do exemplo abaixo, a pasta "*kt*":

O utilizador apenas tem de executar o comando *chmod* para dar permissões de execução do mesmo ao utilizador, como por exemplo um "*chmod 744 Knowledge\ Tester*" (não esquecendo a barra invertida antes do espaço), e o ficheiro ficará com um tom esverdeado e pronto a executar.

Ou seja, não é necessária qualquer instalação para usar o *software*.

Requisitos mínimos

Os requisitos são mínimos em termos de memória ou *CPU*, apenas temos de ter em conta que foi criado para funcionar em processadores de 64 *bits*, pelo que se correr num de 32 falhará, ou seja, as máquinas virtuais criadas terão de ser de 64 *bits*.

O outro requerimento mínimo é obviamente o sistema operativo, tendo sido criado para correr em qualquer máquina *Linux*, e em princípio funcionará em praticamente todas as actuais, pois foi compilado com librarias estáticas incluídas (daí o seu tamanho maior de executável), para que funcione mesmo em máquinas que não tenha as librarias mínimas já instaladas.

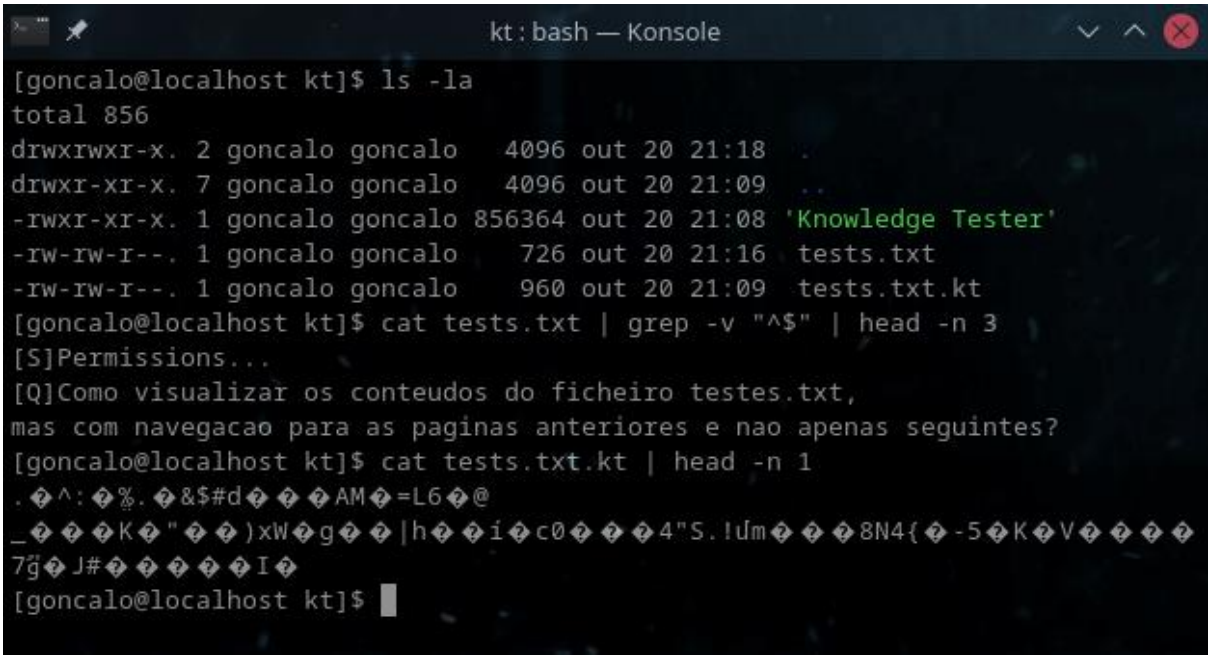
Já foi até colocado a funcionar em máquinas virtuais de *Linux*, com 80MB de memória e 35MB de disco com sucesso.

Tipos de ficheiros do Knowledge Tester

Existem dois tipos de ficheiros principais neste *software* em termos de conteúdo:

- Ficheiros em modo de texto bruto (*raw*);
- Ficheiros em binário (não legíveis);

Os ficheiros de exame em modo de texto, podem ter a extensão que o utilizador desejar, estando no exemplo abaixo com a extensão ".txt" (o ficheiro "*tests.txt*"), enquanto que os ficheiros de exame do *Knowledge Tester* em modo binário, têm a extensão "*.kt*", como o ficheiro "*tests.txt.kt*" abaixo:



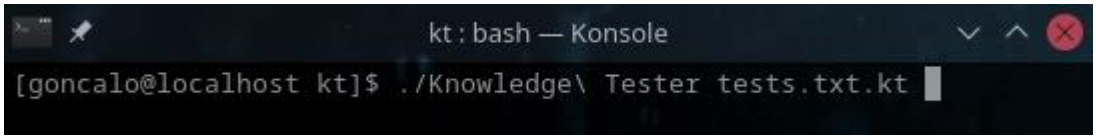
Podemos ver acima que o ficheiro de extensão "*.kt*" é ilegível, pois não está em modo de texto *raw*.

Aprenderemos a criar ficheiros "*.kt*" posteriormente.

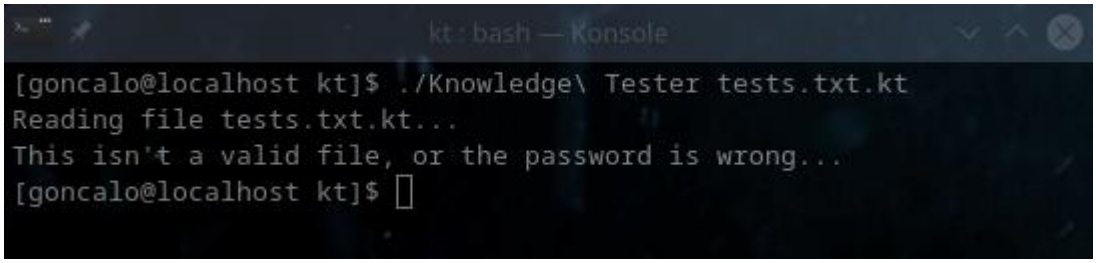
Como executar um exercício ou exame

Se corrermos a aplicação sem nada mais, teremos apenas um alerta de que podemos usar o argumento "*--help*" para obter ajuda sobre como o usar, pois temos de ter no mínimo um argumento, que é o do ficheiro a ser aberto.

Assim, desde que tenhamos um ficheiro de exercícios ou exames criado, só temos de correr a aplicação seguida do nome do dito ficheiro, e a aplicação irá abri-lo de imediato:



Se surgir um erro de "*password is wrong*", é sinal de que o exame está protegido com palavra-passe, e só abrirá se usarmos o argumento "*--password*" seguido da mesma:

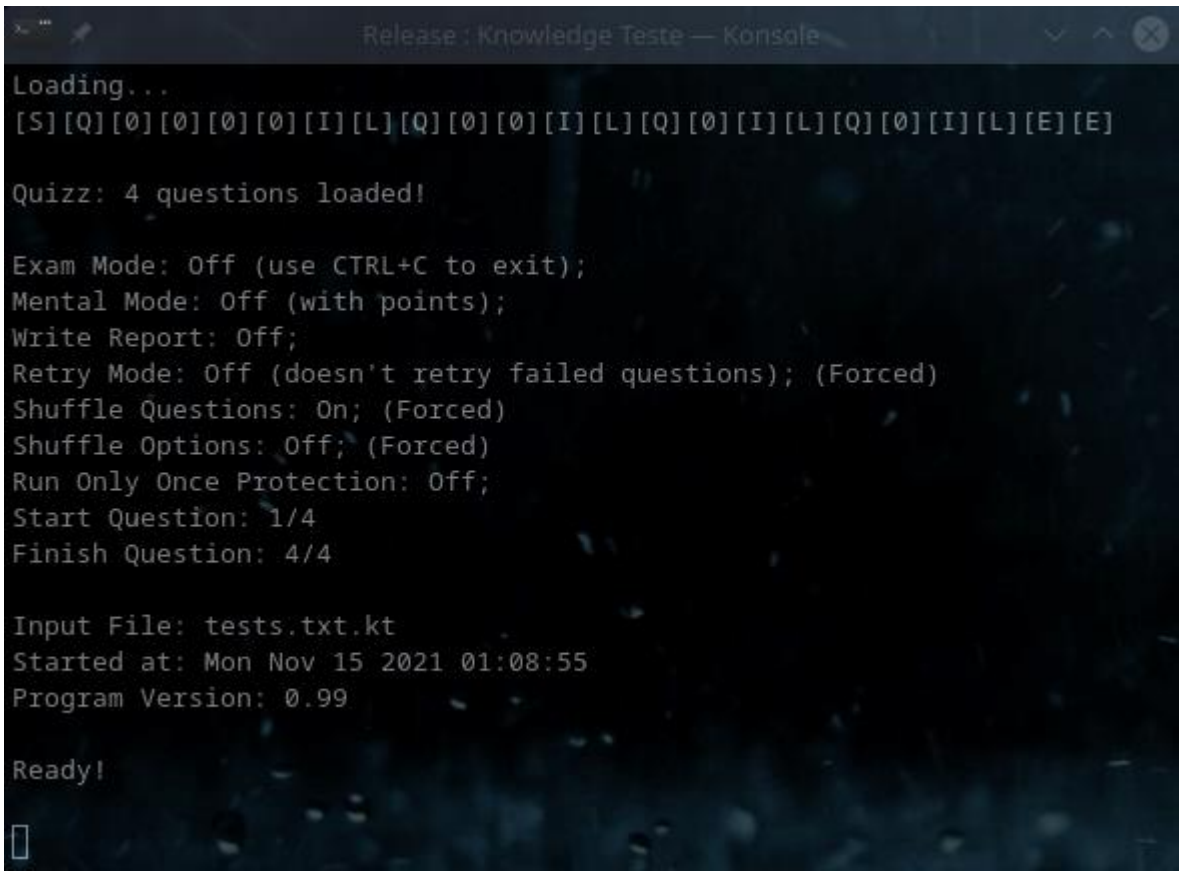


Abaixo podemos ver como abrir um ficheiro de testes, com a *password* fornecida pelo autor do mesmo:

Após ser aberto um exame, surge-nos um ecrã de abertura, onde é exibida (por tradição) a lista de perguntas e respostas, onde podemos ver que algumas têm algo como "[1][0][0][0]" (uma opção certa "[1]" e várias erradas "[0]", nas de escolha múltipla), ou "[W]", que representa uma resposta directa e não de escolha múltipla.

Abaixo vemos as mesmas num arranque em modo "*exam mode*":

Quando no modo "*exam mode*", todas as respostas são escondidas, como podemos ver acima, aparecendo todas como "[0]", enquanto que no modo normal, aparecem visíveis:



Acima podemos ver também que além de nos ser dito quantas questões compõem o exame, é-nos mostrada uma tabela pequena com as opções do exame, como:

- ***Exam Mode***: Este é o modo de exame, e quando activo, não nos permite sair pressionando o atalho "*CTRL+C*", para evitar que um formando/aluno pressionem "*CTRL+C*" quando vêem que a resposta é esse mesmo atalho, e o usem ao invés de escolher o número correspondente (já aconteceu).

  Deste modo, durante o modo de exame, só se consegue sair do programa, escrevendo duas linhas com "*:q!*" nas respostas directas, e ao fim da segunda linha assim, e pressionando *Enter*, o programa termina.

- ***Mental Mode***: Este é um modo de prática rápido, em que o utilizador visualiza na sua cabeça a resposta, e vê se acertou, premindo uma tecla, ao invés de responder para obter pontuação, e será falado mais abaixo.
- ***Write Report***: Este argumento define se será gravado um relatório no fim do exame, o que está activado por defeito quando no modo de exame, mas poderemos desactivá-lo assim nesse modo ou activá-lo noutros modos em que costuma estar desligado por defeito.
- ***Retry Mode***: Neste modo, as perguntas falhadas pelo utilizador, são reenviadas para o fim da lista, pelo que assim, o utilizador irá repetir as perguntas após chegar ao fim do exame, e o mesmo só terminará até ele ter acertado em todas elas.
- ***Shuffle Questions***: Com este modo activo, as perguntas são baralhadas, e assim aparecerão sempre em ordem diferente em cada exame, para evitar que o utilizador sem querer memorize as respostas visualmente, ao invés de aprender a matéria.
- ***Shuffle Options***: Com este modo activo, as opções de cada pergunta, quando forem de resposta múltipla, serão baralhadas sempre que a pergunta aparece, para evitar que o utilizador memorize a resposta visualmente ao invés de aprender a matéria.
- ***Run Only Once Protection***: Com esta protecção ligada, o exame em questão só corre uma única vez, e se tentarmos correr o mesmo exame novamente nessa máquina, não conseguiremos com as protecções activadas, que não são muito avançadas de momento, mas o suficiente para evitar que alunos/formandos tentem correr o mesmo exame duas vezes sem que o formador note.
- ***Start Question* e *Finish Question***: Aqui surgem a primeira questão e a última questão do exame/questionário a serem lidas pelo programa, sendo que todas as anteriores ou seguintes não surgirão no exame.

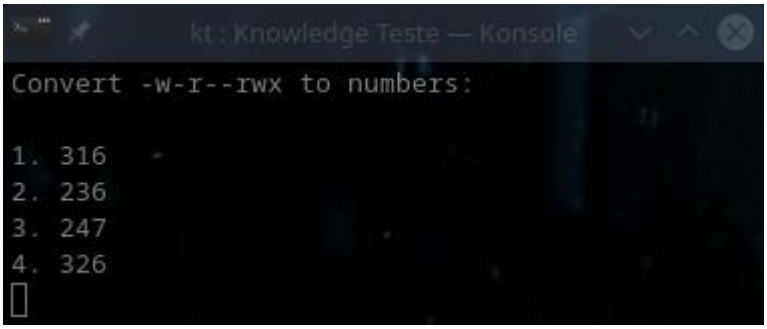
  Estas opções de começo e fim, permitem a que um utilizador possa por exemplo fazer um teste com as perguntas 10 a 20, de um questionário com 100 perguntas, para escolher quais quer treinar, para poder aprender mais rápido.
- ***Input File:*** Aqui aparece obviamente qual é o ficheiro de testes que é lido no exame que está a decorrer.
- ***Started At***: Aqui surge a data correcta de começo do exame, sendo que é importante que o utilizador defina a hora correcta do sistema, para que a data de começo e fim apareça correcta no relatório, com um simples *date -s "20 OCT 2021 22:10:00*".
- ***Program Version***: Este campo define a versão do *software* necessária para correr o exame, sendo que se por exemplo o exame tiver definido com uma versão 1.02, e o *software* for apenas a versão 1.01, o exame não correrá.

E após o "*Ready!*" final do ecrã de abertura, só temos de pressionar uma tecla, e o exame começa.

Normal Mode

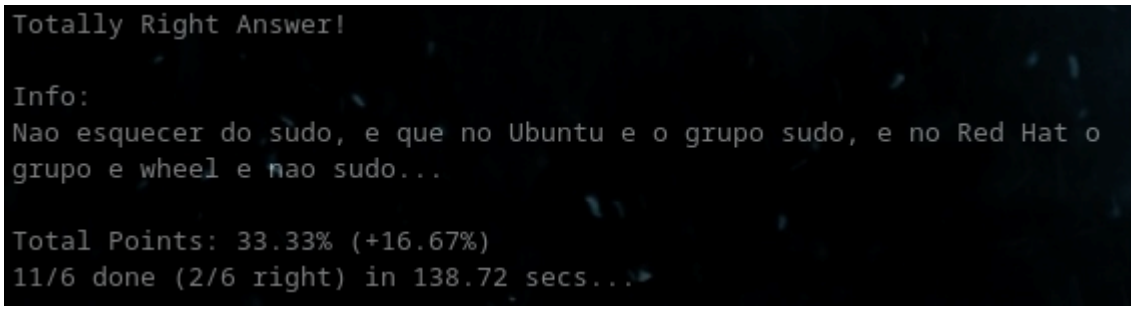
O modo normal é o modo que temos quando não invocamos o argumento "*--examMode*" nem o argumento "*--mentalMode*" ao correr o programa.

No modo normal, tal como no modo de exame, temos um número atribuído a casa resposta, no caso de escolha múltipla, para que ao pressionarmos a tecla desse número, ele nos confirme se acertámos ou falhámos a resposta:



E é semelhante ao modo de exame, no sentido de que temos uma pontuação, mas com algumas diferenças.

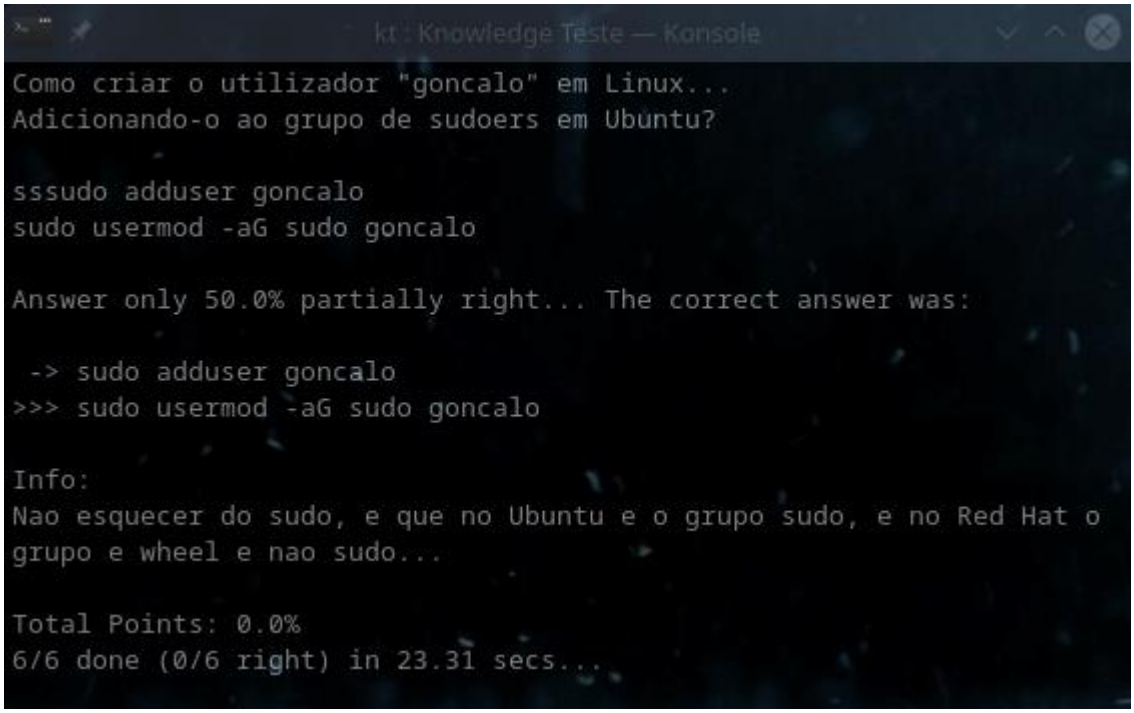
Uma diferença importante ao modo de exame, é que aqui temos as ajudas visíveis, que são os textos definidos com a *tag* "[I]" na criação do exame, e que visam dar uma ajuda ao utilizador sempre que certa ou falha uma resposta, para que ele aprenda:



Podemos ver na imagem acima, que após o título "*Info:*", surge um texto de ajuda ao utilizador, gerado pelo criador do exame. Esta funcionalidade não existe obviamente no modo de exame, ou o utilizador saberia sempre as respostas.

Outra delas, é a de que podemos cancelar o exame a qualquer momento com o atalho "*CTRL+C*", coisa que não acontece no modo de exame, onde tal atalho foi cortado por ter visto um caso de alguém que pressionou "*CTRL+C*" para responder a uma questão em que "*CTRL+C*" era a resposta, ao invés de escolher o número da mesma.

Outra diferença, é a de que se uma pergunta de multi-linhas for respondida apenas parcialmente correcta, no modo de exame ela não voltaria para o fim da lista, e neste modo volta, e o utilizador terá de a repetir as vezes que forem precisas até estar 100% certa, para o teste terminar.

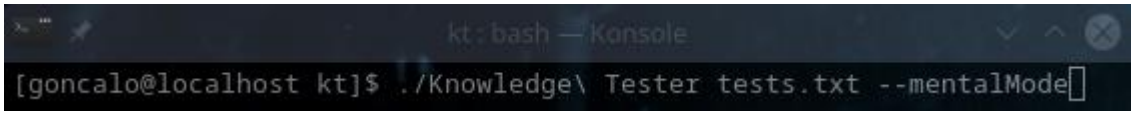


E a principal é a de que no modo exame, se a resposta não for vazia, não poderemos responder mais à mesma, enquanto que aqui, se não usarmos o argumento "*--retryOff*", as perguntas serão enviadas para o fim da lista para as tentarmos novamente, mil e uma vezes, até as acertarmos a 100%.

Mental Mode

Este é um modo de prática rápido, em que o utilizador visualiza na sua cabeça a resposta, e vê se acertou, premindo uma tecla, ao invés de responder para obter pontuação, e nela não só não existe pontuação como não existem teclas a pressionar, basta um simples pressionar da tecla *Enter* (ou qualquer outra), e vemos a resposta, e com outro pressionar, vamos à resposta seguinte.

Ele é invocado ao adicionarmos "*--mentalMode*" à linha de comandos que invoca o exame:

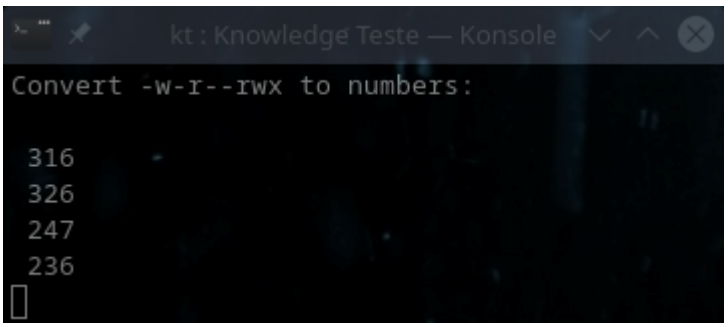


Este modo, permite memorizar muito rápido a matéria, pois sem repararmos, fica tudo gravado nas nossas mentes com as repetições, sem esforço, de forma a que se consigam

aprender uma quantidade enorme de comandos com um esforço mínimo, sem cansaço (se o utilizador não abusar dos exames, claro está).

O objectivo é o utilizador "jogar" enquanto tem vontade, de vez em quando, e sem querer ir aprendendo a matéria, e lendo explicações, e depois praticar no terminal.

No modo mental, não existem opções a escolher:

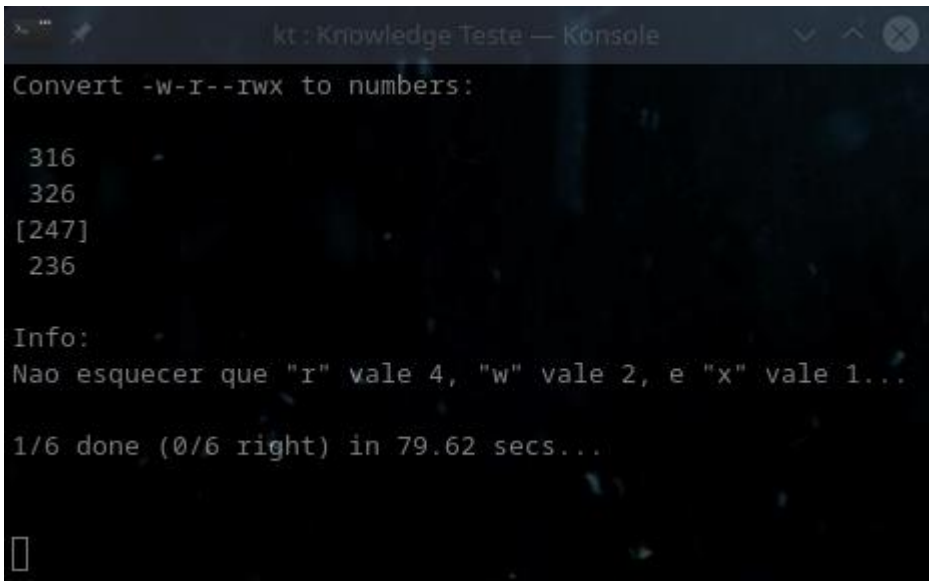


Como se pode ver acima, não há opções para escolher, nem teclas para pressionar, devemos mentalmente pensar qual é a resposta, e pressionar uma tecla qualquer (como por exemplo *Enter*), e a resposta surgirá, e teremos confirmação mental se acertámos ou não.

Isto torna tudo mais rápido, conseguimos responder talvez ao dobro das perguntas na mesma quantidade de tempo, mas tem um senão: como não escolhemos uma resposta por teclas, o programa não sabe se acertámos ou errámos, e por isso não passa as que falhámos para o fim da lista para as repetirmos mais tarde.

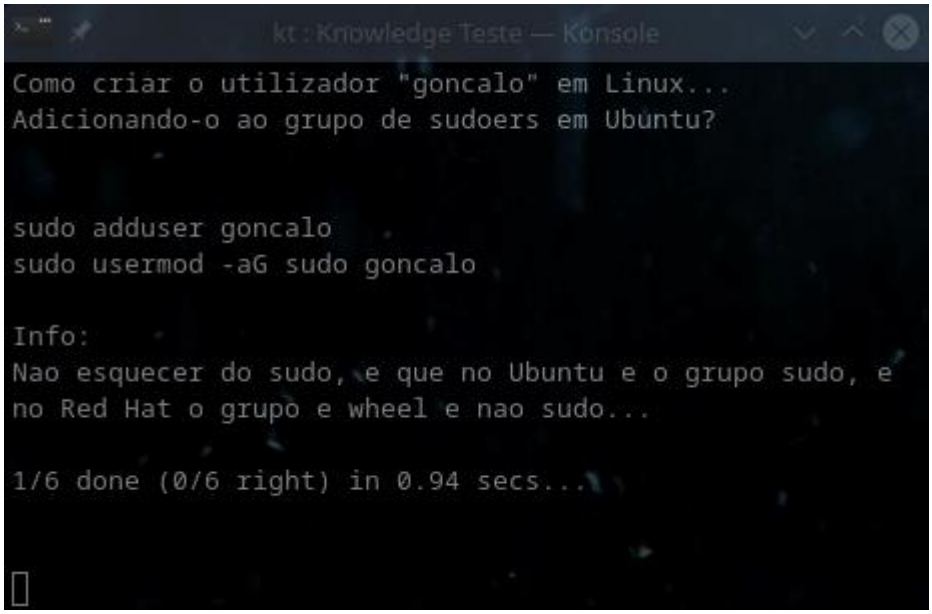
Este modo é especialmente útil se estivermos a estudar para certificações, ou se tivermos muito pouco tempo para aprender algo.

Abaixo podemos ver o que acontece se pressionarmos uma tecla neste modo:



Como podemos ver acima, ao pressionar uma tecla, surgem-nos parêntisis rectos a envolver a resposta certa, para sabermos mentalmente se acertámos na resposta certa ou não.

Também em respostas directas, as mesmas nos aparecem no ecrã com um simples pressionar de qualquer tecla:



Não existe é pontuação, mas é muito bom modo para praticar.

Exam Mode

Este é o modo de exame, e é usado para exames, e tem várias diferenças do modo normal.

Primeiro, só podemos tentar cada pergunta uma única vez, e se a falharmos, não poderemos voltar a tentá-la.

Podemos contudo, pressionar *Enter* numa pergunta (não a responder), e essa pergunta vai para o fim da lista, para a tentarmos mais tarde. Isto é útil para o utilizador deixar para o fim do exame as perguntas mais difíceis que não se lembra, e tentar as que conhece melhor primeiro.

Assim, o utilizador despacha as que sabe, e deixa o tempo do fim para as que não sabe.

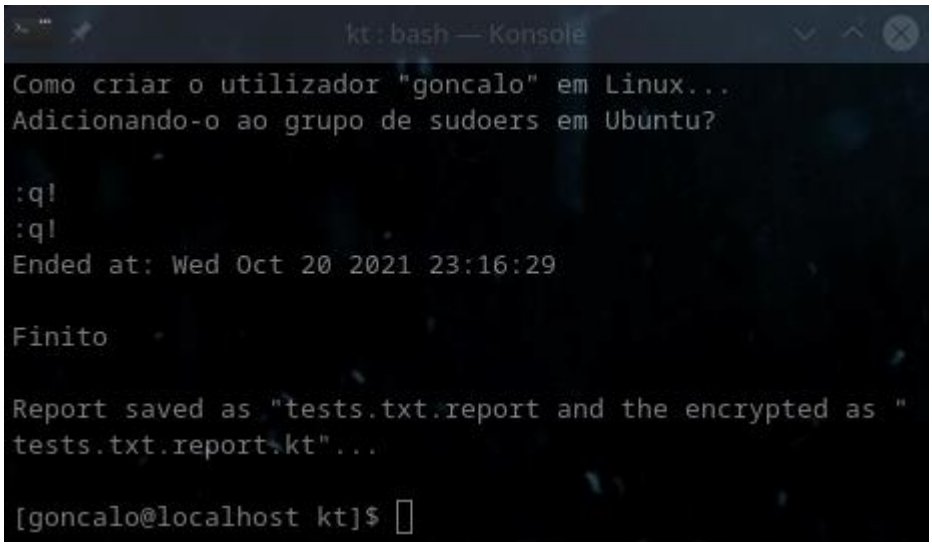
Mas ao contrário do modo normal, aqui o número de perguntas tentadas não sobe para o fim sempre que passamos uma resposta para o fim.

Se o utilizador tentar responder e falhar, não pode voltar a tentar essa questão.

Esta funcionalidade de deixar para o fim questões não respondidas, pode ser anulada com o argumento "*--retryOff*" quando se invoca o exame.

Também, quando activo, este exame não nos permite sair pressionando o atalho "*CTRL+C*", para evitar que um formando/aluno pressionem "*CTRL+C*" quando vêem que a resposta é esse mesmo atalho, e o usem ao invés de escolher o número correspondente (já aconteceu).

Deste modo, durante o modo de exame, só se consegue sair do programa, escrevendo duas linhas com "*:q!*" nas respostas directas, e ao fim da segunda linha assim, e pressionando *Enter*, o programa termina:



Como podemos ver acima, ao escrevermos duas vezes "*:q!*", uma por linha, o exame termina, apesar de ser uma funcionalidade ainda não muito bem testada, e é temporária. Para quem tiver curiosidade, este comando é o mesmo usado no editor de texto *VI*/*VIM*.

Os relatórios de exames

Sempre que termina um exame, são gerados relatórios do mesmo, um desencriptado, em modo de texto puro (*raw*) que o utilizador pode ler e rever, com extensão extra "*.report*", para ver a pontuação que obteve e onde falhou, e um outro encriptado, de extensão "*.report.kt*", sendo que este outro, só pode ser desencriptado pelo autor do exame, com uma *password* que não costuma ser partilhada com os formandos/alunos, sendo que o objectivo é impedir que seja adulterado.

Por norma, são os dois enviados ao examinador, para verificar a pontuação automática, e adicionar pontos extra aos exames, para os casos de quem se enganou por erros de escrita mas que tinha a lógica certa, ou outros pequenos erros, sendo que por norma, o resultado é sempre acima do dado pelo programa.

O parâmetro "*--writeReport*"

Este parâmetro faz com que a geração de relatórios possa ser desligada no modo de exame (onde normalmente está activada, por defeito), ou activá-la noutros modos que não o modo de exame (onde por defeito está desactivada).

Basta adicionarmos então um "*--writeReport*" à linha de comandos que invoca o *Knowledge Tester* para executar um exame, e eles passam a estar activados, ou então um "*--writeReport 0*" ou "*--writeReport false*" à frente de uma linha de comandos que invoque um exame em modo "*--examMode*" para estarem desactivados, mas claro que só funcionará caso não esteja esse parâmetro forçado (algo que será falado noutra parte deste manual).

Como criar um exame

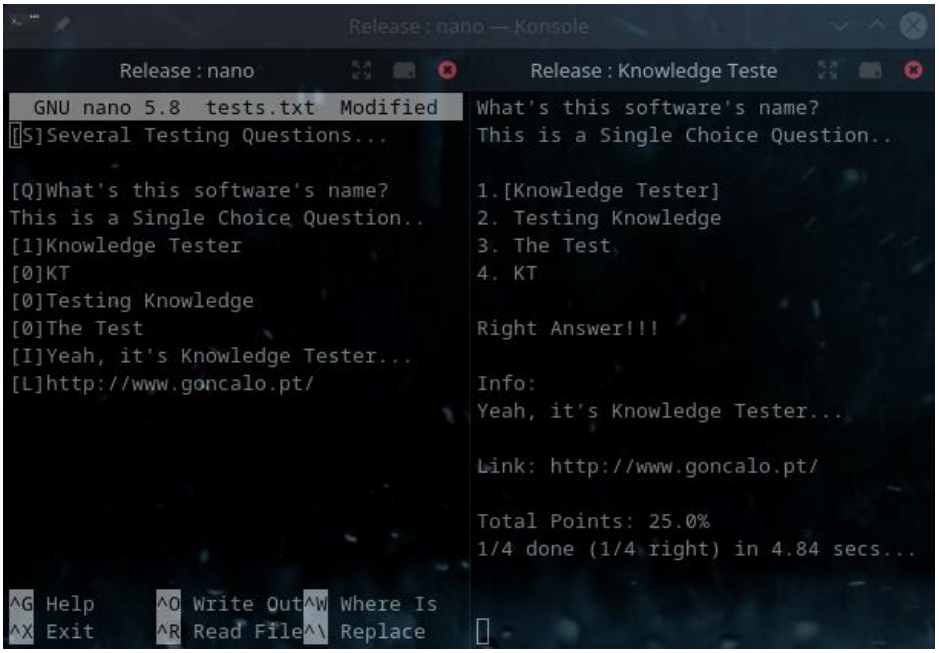
Nos tópicos abaixo, que falam sobre os tipos de perguntas existentes no *software* "*Knowledge Tester*", poderão ver como criar exames.

Não é complicado, só têm de usar algumas *tags* para definir perguntas, respostas, secções, e outras coisas, e são explicadas com mais pormenor nos tópicos seguintes.

Mas antes deixa-se um pequeno resumo das *tags* que se podem usar:

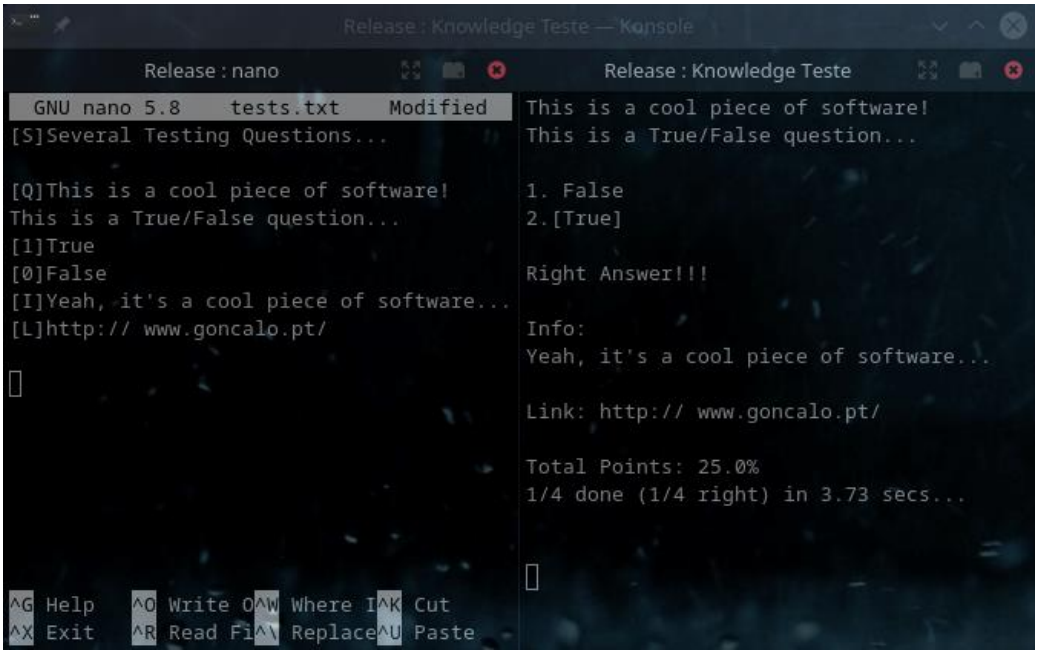
- ***[S]*** - ***Section*** – Esta tag é usada para definir uma secção, que de momento não é muito usada, mas que um dia será usada para permitir a que um ficheiro possa ter muitas secções distintas, e que podemos correr perguntas apenas de uma específica;
- ***[Q] – Question*** – Aqui definimos uma questão, ou seja, o texto que aparece para cada pergunta em si;
- ***[1] – Right Answer*** – Esta *tag* define uma resposta certa, pelo que se uma linha tiver esta *tag* antes, será a resposta certa, quando usamos perguntas "de cruzinhas", onde surgem várias erradas e uma certa e temos de escolher a certa. De momento o *software* não permite escolha múltipla, pelo que só uma poderá ser usada por questão;
- ***[0] – Wrong Answer*** – Com esta *tag* definimos as respostas erradas, ou seja, todas as respostas que as tenham, serão erradas e farão o utilizador perder a oportunidade de ganhar pontos se as escolherem;
    - **NOTA:** É de notar que neste tipo de respostas, seja de escolha por cruzes (uma *tag* ***[1]*** e várias ***[0]***), seja em respostas de verdadeiro ou falso (quando temos apenas duas respostas, a verdadeira com "***[1] –*** Verdadeiro", e a falsa com "***[0]*** – Falso"), basta-nos escolher a resposta pressionando as teclas de 1 a 9, pois temos um limite de momento de 9 opções possíveis, e ao ser pressionada a tecla, a resposta é automaticamente enviada, por isso há que ter cuidado ao responder. É de notar que responder com os números do teclado numérico também não é garantido, terá de ser com as teclas normais de números, as que estão por cima das letras no teclado.

      Podemos ver abaixo como criar uma pergunta de escolha única:



Notem que a ordem inicial das questões não importa muito, porque por defeito a aplicação coloca as opções no ecrã de forma aleatória, mas o que podem ver acima é o suficiente para se criar um exame passível de ser executado.

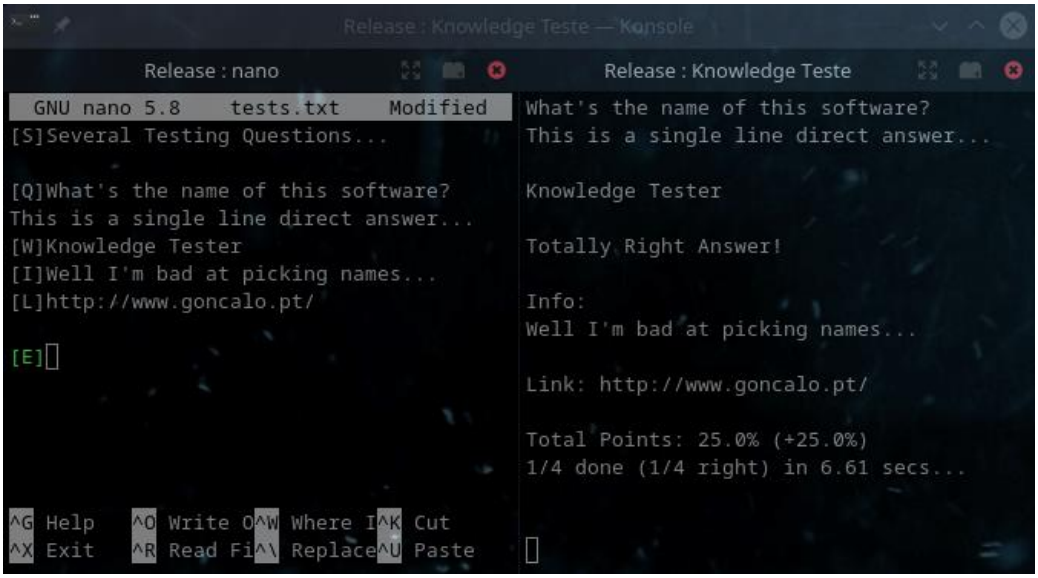
Podemos ver agora como criar uma pergunta de verdadeiro ou falso:



- ***[W] – Direct Answer*** – Quando uma ou mais linhas de texto estão precedidas por esta *tag*, será uma resposta de escrita, pelo que o utilizador tem a oportunidade de escrever frases inteiras de respostas antes de as submeter com a tecla "*Enter*", e a

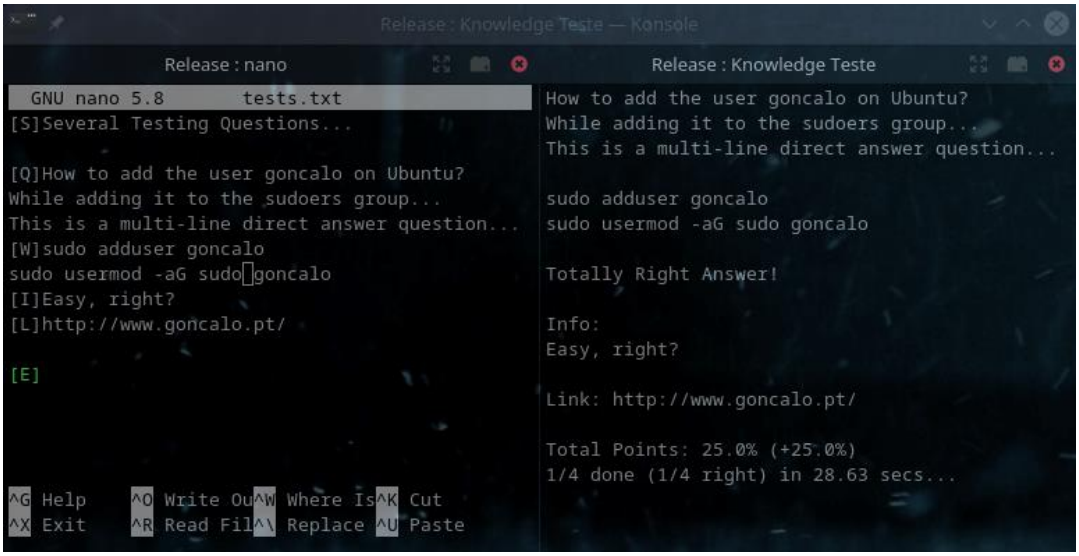
resposta só é finalmente submetida, quando submetemos uma linha completamente vazia, pelo que podemos submeter respostas de múltiplas linhas, quando estamos a por exemplo responder a laboratórios de múltiplas linhas de comandos;

- Abaixo podemos ver como criar uma pergunta de resposta directa:



Podemos ver acima que basta colocar a resposta após uma *tag* "[W]" que ela é definida.

Abaixo podemos ver como definir uma pergunta de resposta directa multi-linha:



- ***[I] - Info*** – Aqui definimos uma pequena ajuda, ou seja, quando o utilizador responde, verá após a sua resposta um texto com esta ajuda (quando não estamos no modo de exame claro está), que nos ajudará a aprender mais sobre a resposta, porque deveria

ter sido respondido de certa maneira, etc, o que faz com que este *software* seja útil não só para praticar como também para aprender a matéria, em simultâneo.

- ***[L] – Link*** – Após esta *tag*, Temos um endereço para um *website* ou página na *Internet* com mais dados sobre o assunto, isto já a pensar nalguma versão com interface gráfica que vá ser desenvolvida um dia, mas também relembrando que em várias aplicações de terminal, como a *Konsole* do *KDE*, podemos clicar no *link* mesmo na *shell* com o rato, e o *link* abrirá no *browser*.
- ***[E] – End*** – Esta *tag* define o fim do ficheiro de perguntas, pelo que tudo o que vier após esta *tag* será ignorado pelo *software*, apesar de esta *tag* ser opcional e os exames poderem ser executados sem ela.

Assim, basta criar um ficheiro de perguntas, obedecendo a estas regras, e depois correr os exames, e tudo funcionará.

Os tipos de perguntas

De momento existem apenas dois tipos de questões principais, as de escolha múltipla e resposta directa, e que se dividem no fundo em quadro:

- **Escolha Múltipla:** Nesta são apresentadas várias opções ao utilizador, estando apenas uma certa, e tendo de escolher a opção certa.
- **Verdadeiro ou Falso:** Esta é uma variante da de escolha múltipla, e no fundo consiste numa pergunta, e temos como resposta apenas “Verdadeiro” ou “Falso”, sendo que uma tem “[1]” antes e a outra “[0]” quando é criada. Desta forma, estamos a ter uma pergunta de Verdadeiro ou Falso, com escolha múltipla.
- **Resposta Directa:** Aqui é-nos dada a oportunidade de responder por escrito a uma pergunta, e ser-nos-á depois confirmado se a resposta está certa ou errada.
- **Laboratório (Resposta Directa Multi-Linha):** Esta é uma variante da resposta directa, mas com uma resposta multi-linha, onde o sistema depois verifica quais as linhas em que acertámos da resposta original, para nos gerar uma pontuação.

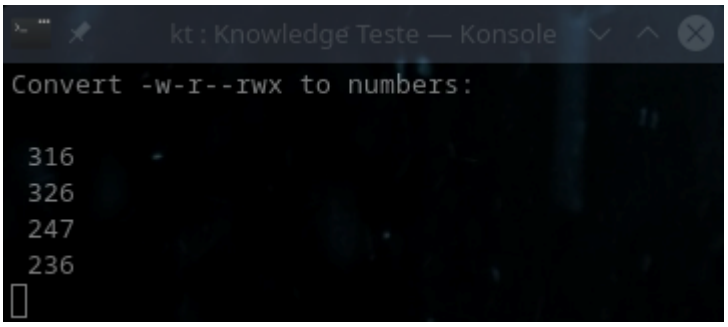
Serão adicionadas novas funcionalidades mais tarde, mas estas são as mais importantes.

Temos agora um resumo abaixo de cada uma em separado.

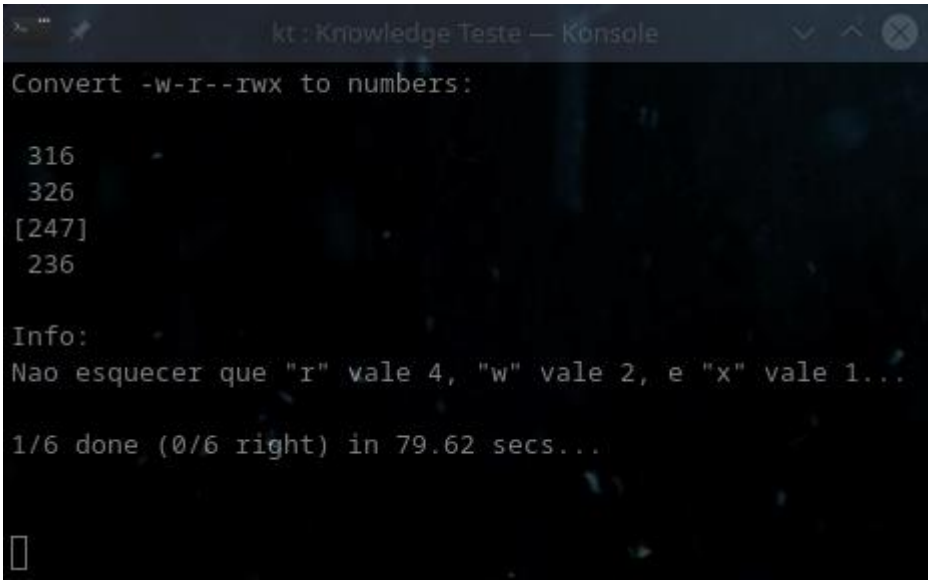
Escolha Múltipla

Nesta variante de perguntas, são apresentadas várias opções ao utilizador, estando apenas uma certa, e tendo de escolher a opção certa.

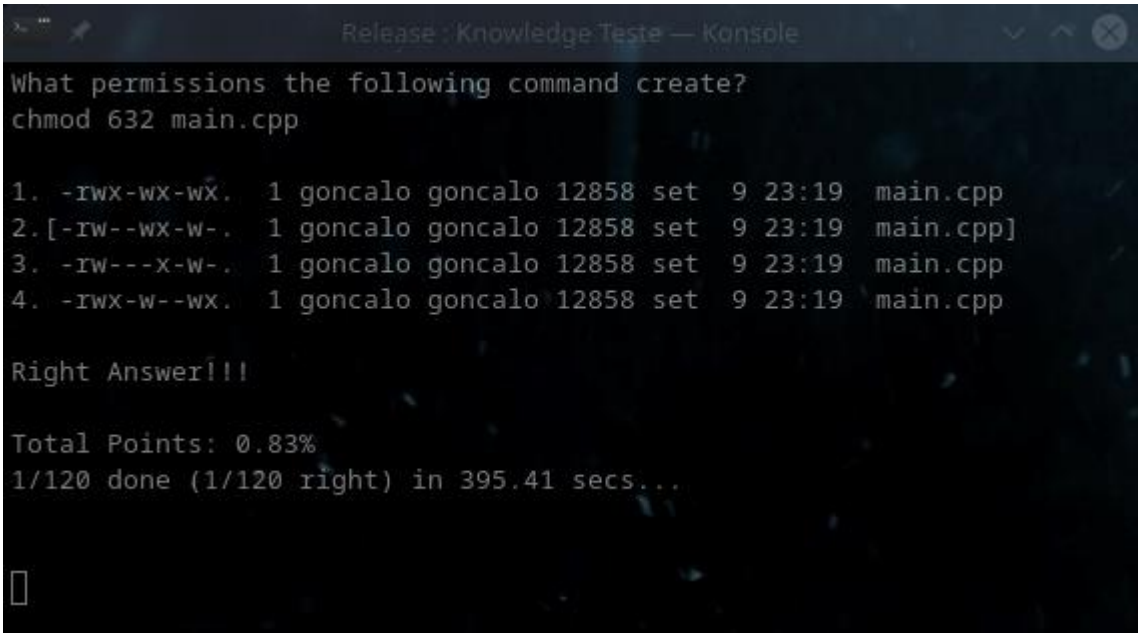
Abaixo temos como exemplo uma pergunta no modo "*--mentalMode*", em que não aparecem as opções numéricas para clicarmos para escolher a questão, onde nos basta apenas pressionar uma tecla para visualizar se acertámos no resultado ou não:



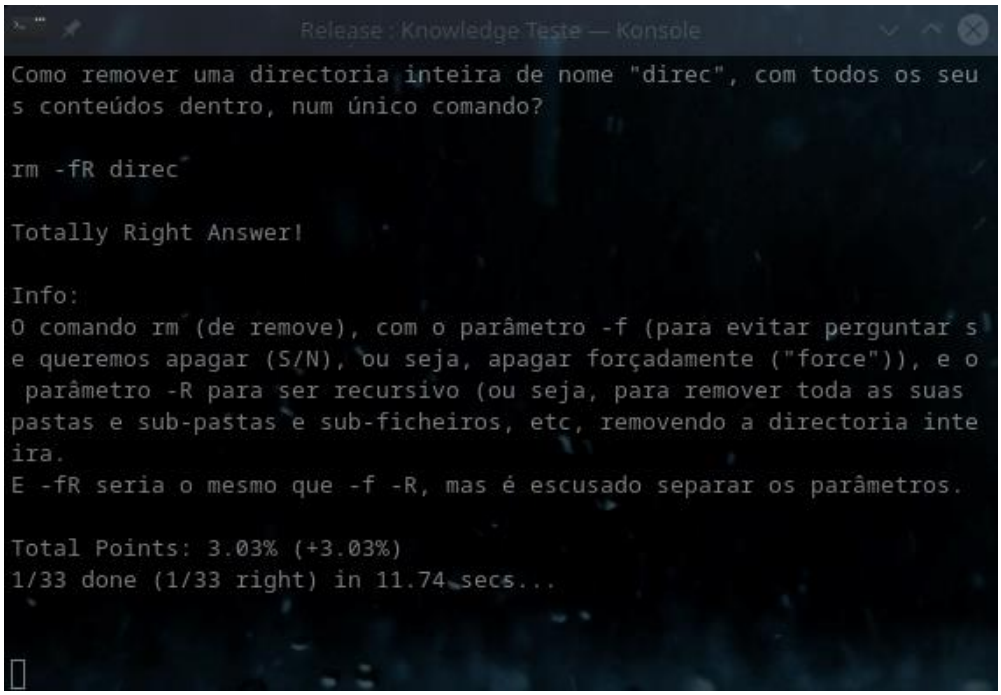
Ao clicarmos numa tecla, a opção certa ganha realce e ficamos a saber se acertámos ou não, mentalmente, pois o *software* não saberá pois não tem *feedback* por parte do utilizador, pois ele se limita a clicar numa tecla para prosseguir:



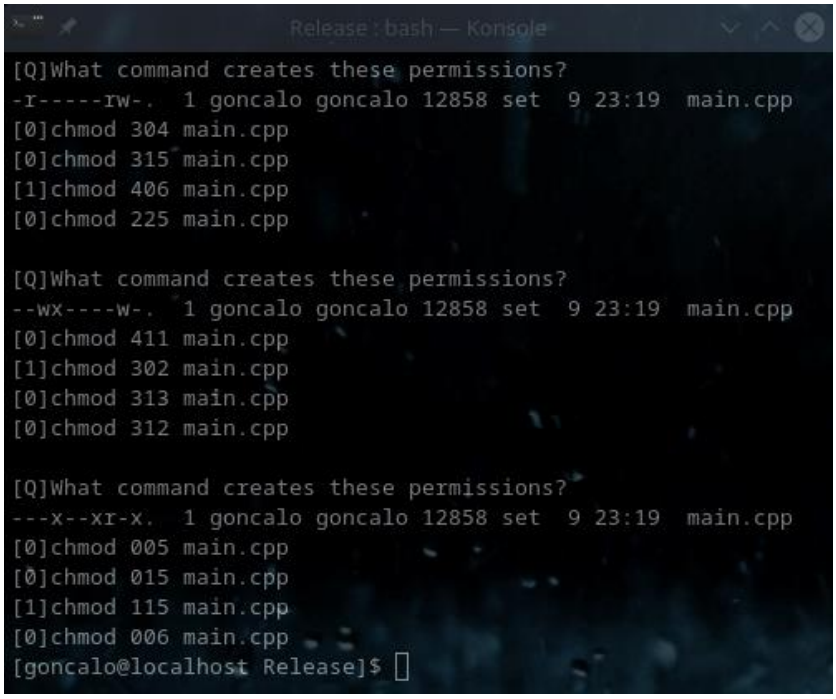
No modo de exame ou no modo normal, já aparecem as opções para escolha e as respostas já são avaliadas:



E quando respondemos a alguma pergunta que tenha ajuda, aparece-nos essa ajuda sobre a questão, além da informação sobre se está certa ou errada:



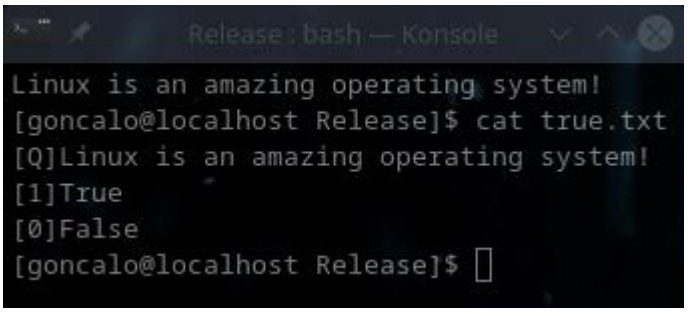
Estas perguntas de escolha múltipla são criadas de forma fácil, com uma *tag* "[0]" a representar as respostas erradas, e uma "[1]" a representar a resposta certa, e o programa trata depois de baralhar as opções:



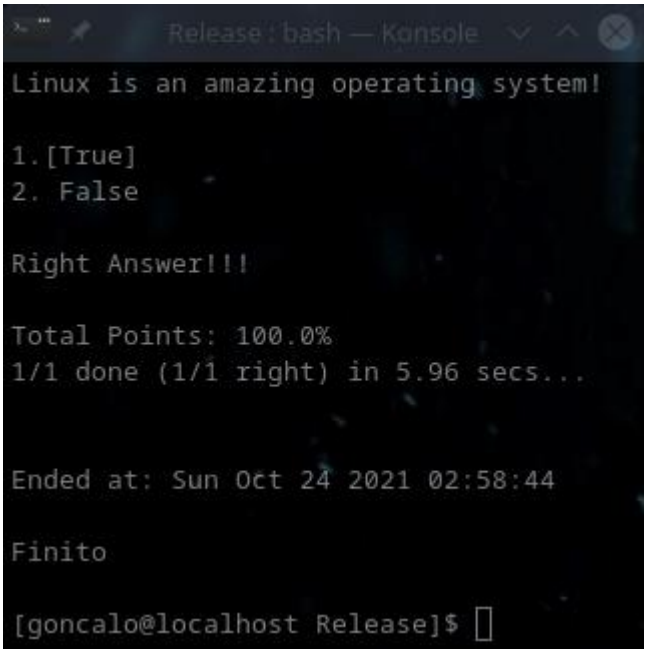
As respostas acima são geradas dinamicamente com a funcionalidade do parâmetro "*--createFilePermissions*".

Verdadeiro/Falso

Esta é uma variante da de escolha múltipla, e no fundo consiste numa pergunta, e temos como resposta apenas "Verdadeiro" ou "Falso", sendo que uma tem "[1]" antes e a outra "[0]" quando é criada. Desta forma, estamos a ter uma pergunta de Verdadeiro ou Falso, com escolha múltipla:



A configuração acima, depois dá-nos um exame com perguntas simplificadas de "Verdadeiro" ou "Falso", onde no fundo são apenas duas respostas num teste "de cruzinhas".

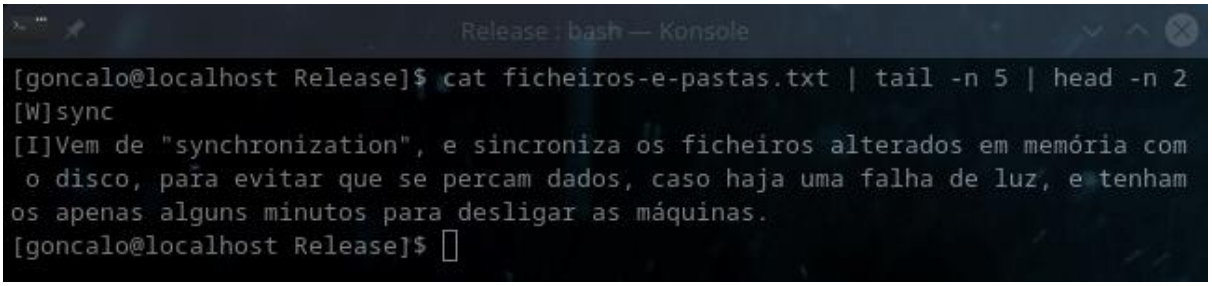


Resposta directa

Esta é uma funcionalidade que testa melhor os nossos conhecimentos, pois obriga-nos a escrever a resposta por inteiro, e é útil na memorização por exemplo de comandos de Linux.

Aqui é-nos dada a oportunidade de responder por escrito a uma pergunta, e ser-nos-á depois confirmado se a resposta está certa ou errada.

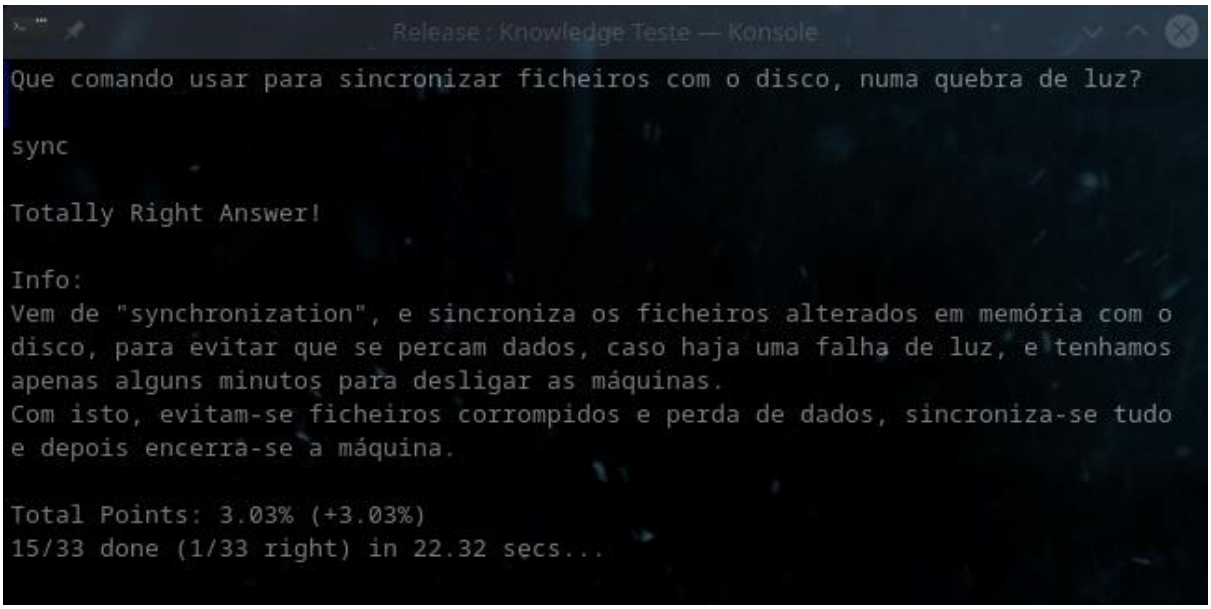
Ela é criada substituindo a *tag* certa de "[1]" por "[W]", de "*written*":



Ela dá-nos depois também a informação sobre se está a resposta correcta ou não, bem como a informação sobre a mesma, criada com a tag "[I]" aquando da criação do exame.

Este tipo de perguntas é fácil de criar, basta que no editor coloquemos uma *tag* "[W]" (de "*written answer*") ao invés das típicas [1], para a resposta certa, e basta essa única *tag*.

E a resposta será dada directamente, por escrito:

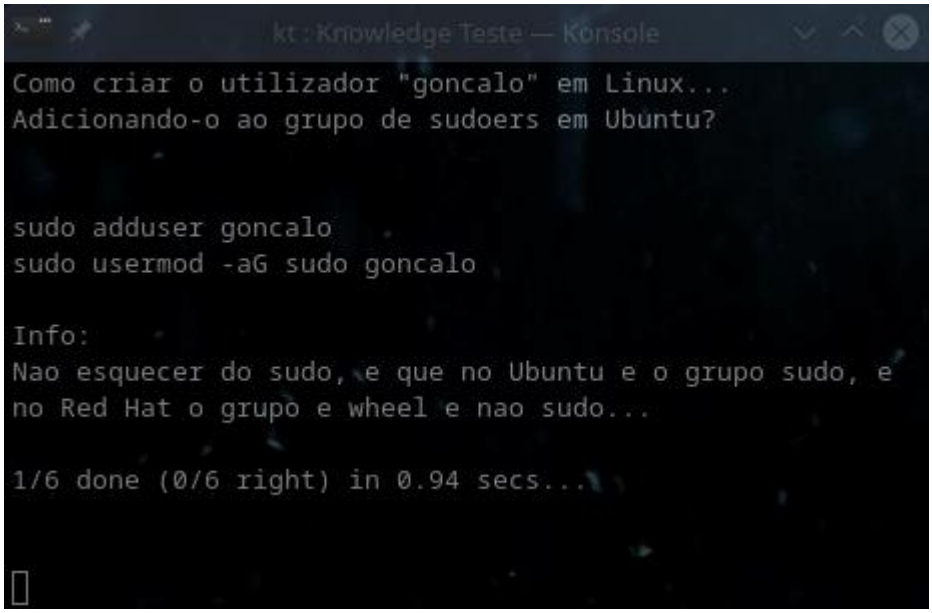


O *software* está preparado para identificar respostas mesmo que tenham um número diferente de espaços entre palavras, mas se um caractere que não espaços estiver errado, a resposta será marcada como errada e não contabilizada na pontuação, e caberá ao examinador depois rever o relatório gerado no fim do exame para corrigir a pontuação, pelo que por norma, a pontuação do aluno será sempre superior após revisão do examinador, que terá de analisar também erros de escrita, não perceptíveis pelo *software*.

Laboratório (Resposta Directa Multi-Linha)

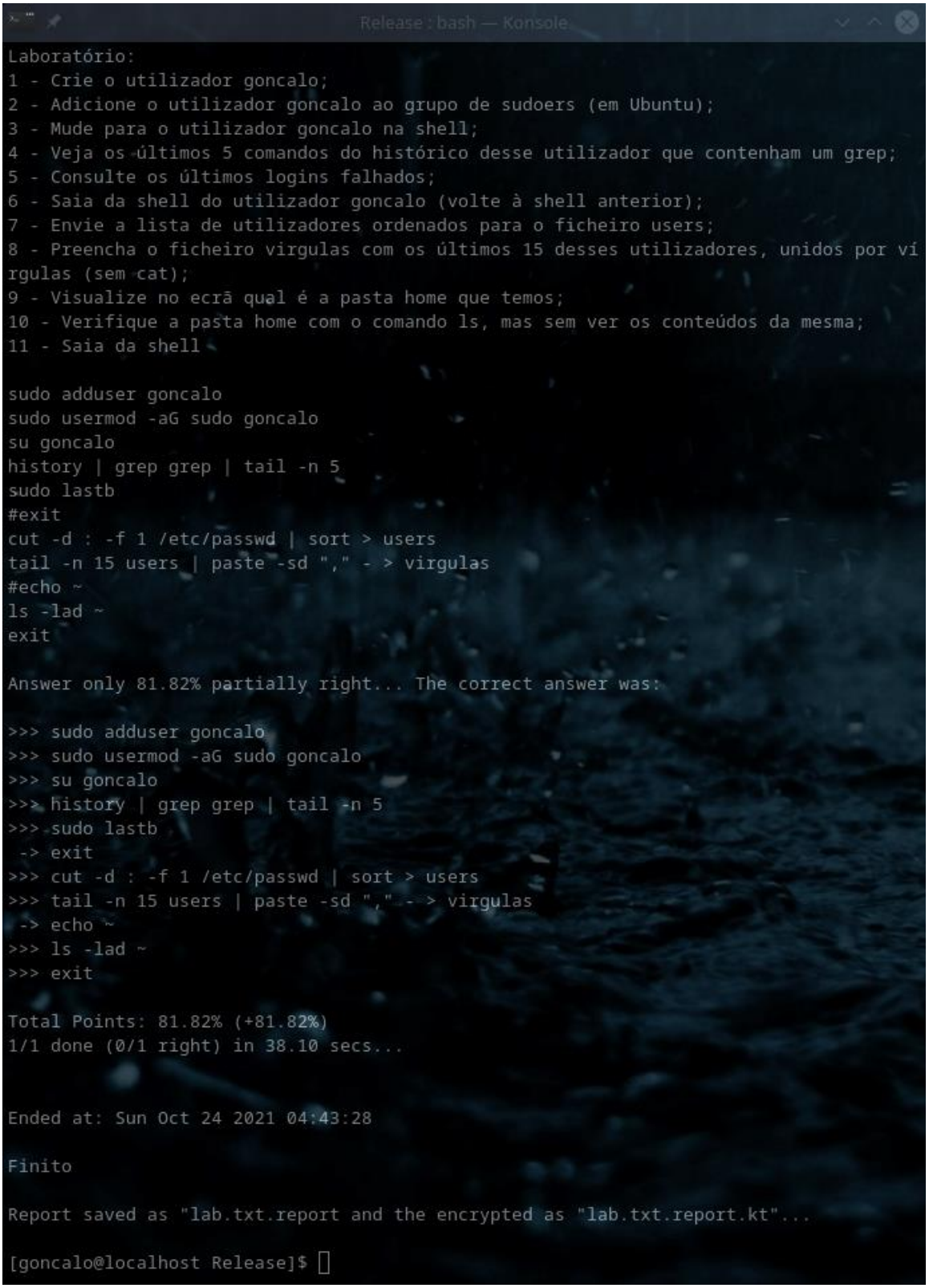
Esta é uma variante da resposta directa, mas com uma resposta multi-linha, onde o sistema depois verifica quais as linhas em que acertámos da resposta original, para nos gerar uma pontuação.

Na imagem abaixo vemos uma resposta a uma pergunta multi-linha, no *mental mode*, para memorização:



Este modo é especialmente útil no treino para certificações com laboratórios, em que teremos de saber de cor processos inteiros com múltiplas linhas de comandos, como certificações *Red Hat*, ou até *Cisco*, entre outras, ou se quisermos pura e simplesmente não nos esquecer de como fazer certas operações que raramente executamos mas que devemos ter memorizadas.

Quando no modo normal ou de exame, já nos é pontuada a resposta multi-linha, onde nos é indicado quais das linhas da resposta estavam certas, e a pontuação da pergunta será dividida entre o número de linhas e a pontuação final contabilizada:



Se virem bem na respostas acima, quando as linhas diferem, aparecem com apenas "->" antes da resposta que deveria ser certa, enquanto que as que estão iguais às respostas certas, aparecem com ">>>" antes.

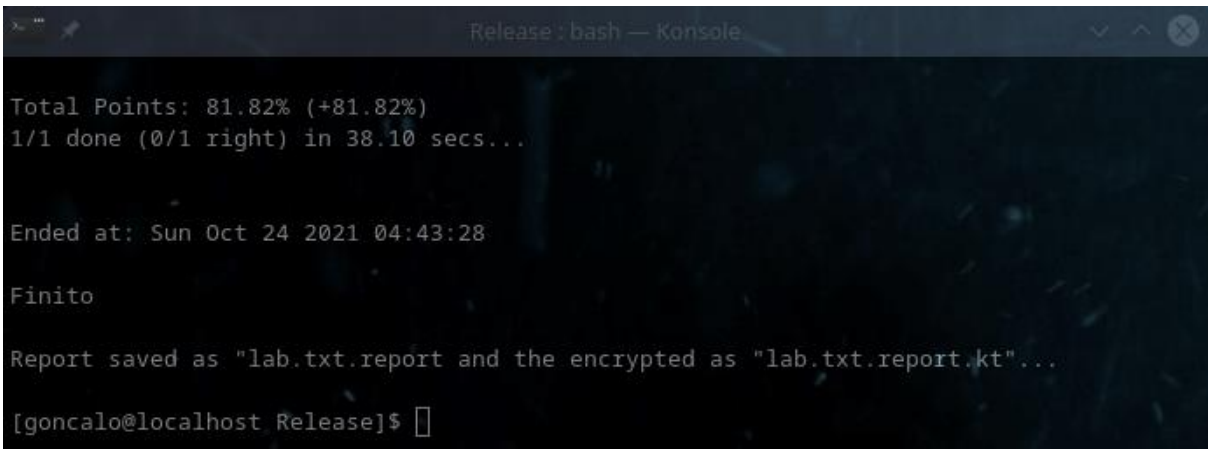
Pontuações

Também é dividida a pontuação total da resposta pelo número de linhas, para se calcular a pontuação final de acordo com o número de linhas que estavam correctas, sendo que neste caso como só existia uma pergunta (era um laboratório de exercício único), ela valeria 100%, e assim, a pontuação final ficou como 81,82% (9 em 11 linhas certas).

Os relatórios dos exames

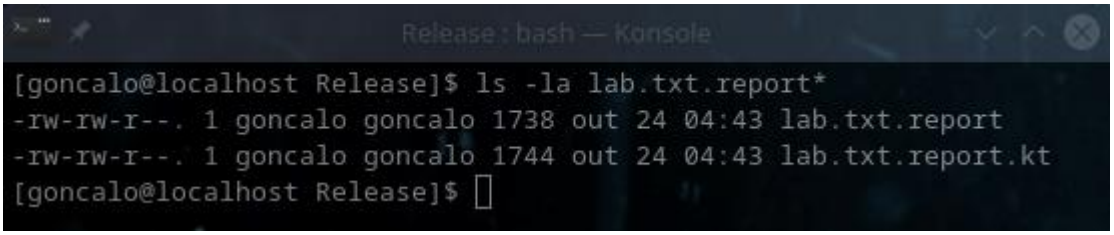
Os relatórios dos exames são extremamente importantes em avaliações.

Não teria sentido alguém ser avaliado sem um relatório exibindo todas as perguntas respondidas, não só porque o formador tem de rever as perguntas falhadas que podem ser erros de escrita, ou pequenas trocas mas onde o estudante mostrou conhecer a lógica por detrás do que queria fazer, mas também para evitar que notas e respostas sejam adulteradas por algum examinado, e em especial aquando de formações à distância:



Sendo assim, sempre que é terminado um exame, são gerados dois tipos de relatórios diferentes, um ficheiro com o nome do exame e extensão "*.report*", e outro com o nome do exame e extensão "*.report.kt*", sendo que o primeiro está em texto bruto ("*raw*"), legível por todos, e que permite ao examinado ficar com uma cópia para ver onde falhou e onde tem de melhorar no Futuro, ou até saber que deve dizer ao formador que numa pergunta qualquer errou por algum motivo, e a segunda encriptada, onde apenas o examinador conseguirá desencriptar e ter acesso ao relatório.

Ambas devem ser enviadas ao formador. A encriptada porque é a válida e inadulterável, e a de texto, só para a eventualidade de acontecer algo de errado, ou algum outro motivo:



Como podem ver acima, há dois tipos de relatórios diferentes, e o de extensão "*.report.kt*" é encriptado e por isso tem uns bytes a mais, típico do processo de encriptação usado neste caso, um sistema de encriptação baseado no algoritmo *Rijndael* conhecido hoje em dia por *AES 256*.

Abaixo podem ver o conteúdo de ambos, e poderão reparar que o que está em modo de texto é visível ao aluno, que poderá levar para casa para ver onde falhou e acertou, e que tem literalmente todo o conteúdo que foi apresentado no ecrã durante o exame, sem excepção, tendo até o menu de entrada, opções escolhidas, perguntas e respostas, pontuação, mesmo tudo. Tudo o que foi apresentado no ecrã durante o exame, estará lá contido.

Abaixo podem espreitar o conteúdo dos ficheiros:

E como podem ver acima, um está em modo de texto *raw*, e o outro encriptado, pelo que contém código binário.

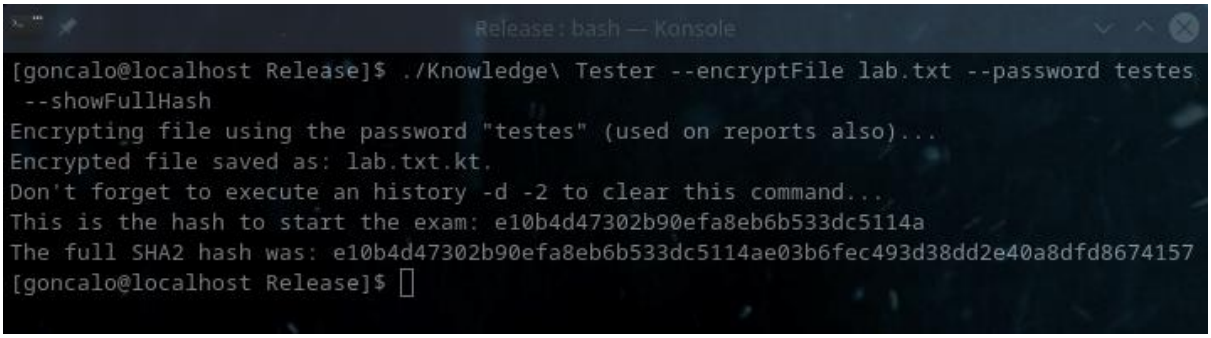
Exames Protegidos

Como proteger um exame

Assim que criamos um exame, podemos encriptar o mesmo com o parâmetro "*--encryptFile*" seguido do nome do ficheiro, junto com o parâmetro "*--password*" seguido da password que pretendemos usar para a encriptação:

Escusado será dizer que para termos protecção total, devemos usar passwords com muito mais caracteres que esta, com símbolos à mistura e não apenas letras, e que não contenham só palavras de dicionário como a "testes" usada acima, pois se alguém tentar um ataque inicial de *brute-force* a este ficheiro encriptado, e experimentar fazê-lo com um dicionário pequeno onde esteja a palavra "testes" contida, ele poderia ser desencriptado não em milhões de anos mas em minutos ou horas.

Pode ser usado também o parâmetro "*--showFullHash*" durante a encriptação, para visualizarem a *hash* inteira original, ao invés de apenas a cortada, e tal será falado posteriormente neste manual:



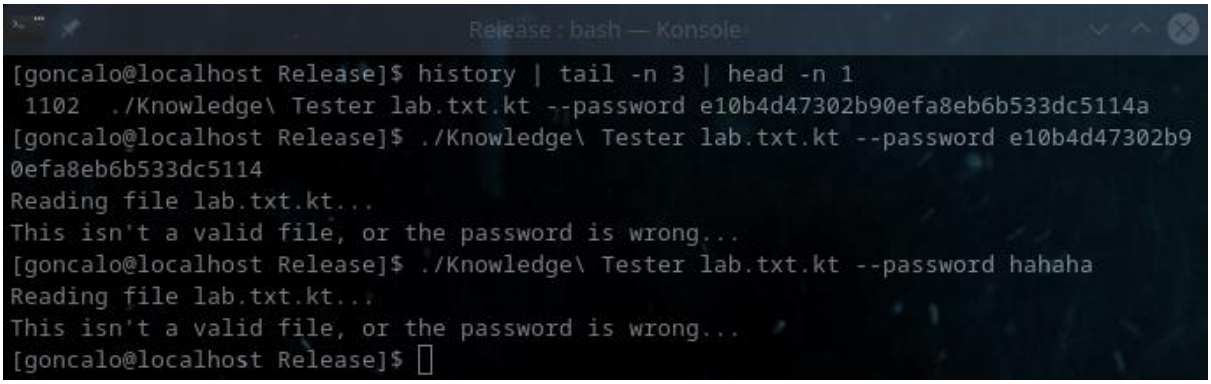
Falamos mais sobre a força das *passwords* no tópico seguinte mais abaixo.

Como correr um exame protegido

Notem na imagem anterior que nos é dada uma hash para se poder iniciar o exame, que no exemplo acima é "*e10b4d47302b90efa8eb6b533dc5114a*", e que será a mesma se tentarem encriptar alguma coisa com a *password* "testes" nas vossas máquinas também, e que é encurtada a 32 caracteres para ser mais fácil de transmitir, mas que não deixa de ser mais do que suficientemente segura.

Esta é a *password* que deverão dar aos formandos/alunos para eles iniciarem os exames. Desta forma eles conseguem iniciar o exame com a password dada, mas não conseguem desencriptar os relatórios com a mesma.

Vejamos como se corre um exame com essa *password*:



No exemplo acima podem ver que corri antes (sem problemas) o exame da maneira correcta, com a palavra-passe que me foi dada inicialmente ao encriptar o exame (a "*e10b4d47302b90efa8eb6b533dc5114a*"), e correu sem sucesso, e é importante referir que qualquer tipo de *hash* usada por este *software* tem força mínima de *SHA2* e devidamente "salteada" para evitar ser descoberta a origem através de *rainbow tables*.

Mas de seguida cortei apenas um caractere a essa palavra-passe (o "*a*" final), e já falhou, com a mensagem "*This isn't a valid file, or the password is wrong...*", e se tentarmos também outra palavra qualquer, como o "*hahaha*" acima, falhará também.

O exame só é aberto na hora certa, quando o formador assim quiser (quando lhes der a *password*), e com a *password* certa.

O exame também corre com a *password* original usada pelo formador, ou seja, o formador pode usar a *password* "testes" e dá-la aos alunos e eles correrem o exame com ela, desta forma todos sabem a *password* original dentro da turma, mas fora da turma ninguém a saberá. O único problema é que desta forma, qualquer aluno poderá adulterar um ficheiro de relatório de exames, apesar de ter de saber fazê-lo.

Há uma opção de fazer com que os exames encriptados com *password* possam ser executados/abertos sem qualquer *password*, que será explicado abaixo.

Como fazer exames encriptados abrir sem *password*?

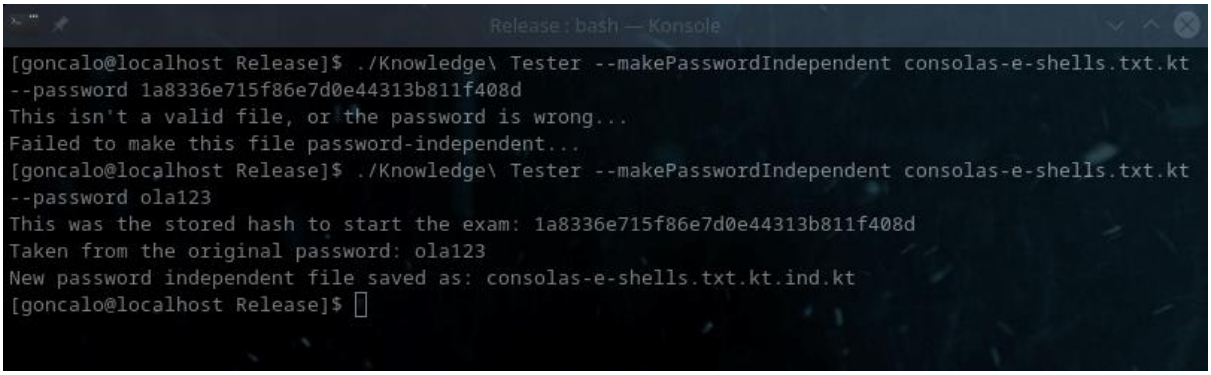
Com a opção "*--makePasswordIndependent*", podemos fazer com que um teste/exame encriptado com *password*, possa ser aberto sem qualquer necessidade de uso de *password*.

Ele continuará encriptado com a *password* usada originalmente, o que significa que exames executados com esse ficheiro, gerarão relatórios encriptados que só poderão ser visualizados/desencriptados com a *password* original (e não a dada aos formandos para execução), e significa também que o exame em si só poderá ser também desencriptado na mesma com a *password* original, que nunca será guardada no ficheiro em si.

Mesmo assim, bastar-nos-á executar "*./Knowledge\ Tester ficheiro.txt.kt*" para o exame abrir, sem ter de saber a *password* original.

Isto é útil quando temos um exame para dar a formandos, que não podem desencriptar para ver as suas perguntas e respostas, nem para verem os relatórios gerados, mas ao mesmo tempo não queremos obrigá-los a ter de colocar *passwords* para aceder.

Assim basta executarem e já está, sem deixar de estar protegido:



Acima podemos ver que só com a *password* original se pode executar esta operação, senão obviamente ganhava-se acesso ao ficheiro mesmo sem ter a mesma.

O ficheiro final acaba por ter adicionada a extensão "*.ind.kt*", de "independente", que poderá ser obviamente alterada mais tarde para um nome mais fácil de memorizar ou usar.

A partir desta operação nunca mais será pedida uma *password* para se abrir este ficheiro.

Atenção que depois, tal como está feito o *software* de momento, não podemos forçar argumentos ou outras operações neste ficheiro, por ser já *stand-alone*, sendo que tais operações têm de ser feitas antes de se tornar *stand-alone*, como por exemplo a "*--forceArguments*" ou algumas outras.

Não fica gravada no próprio exame a *password* original?

Não. O sistema de encriptação deste *software* foi pensado de forma a que um exame quando é corrido, saiba precisamente como gerar um relatório que possa ser desencriptado apenas com a *password* original, sem nunca a saber sequer.

Isto pode parecer confuso a quem nunca programou algo com encriptações, mas é possível.

Isto significa que o relatório tem informação suficiente para gerar um relatório que só possa ser aberto com a *password* "testes", mas ninguém conseguirá nunca encontrar nesse relatório a própria *password* "testes".

Escapa ao âmbito deste manual explicar como tal é feito, mas pode-se garantir que a *password* original usada pelo formador para gerar um exame, nunca sairá das suas mãos a qualquer momento, e só com a mesma é que um relatório poderá ser desencriptado.

É importante referir assim, que essa *password* usada pelo formador para gerar exames, nunca é guardada em lado nenhum, nem sequer no exame, nem noutro lado qualquer, a nenhum momento, sendo que se o formador a usar, e correr o comando "*history -d -2*" logo de seguida para eliminar a mesma do histórico da *shell Linux* que está a usar, ela desaparece de vez e ficará apenas guardada na sua mente.

Que *password* têm exames encriptados sem *password*?

Um exame encriptado sem o uso do parâmetro "*--password*" seguido da palavra-passe, poderá ser aberto sem o uso de qualquer tipo de *password*, ele estará encriptado com a *password* por defeito do *Knowledge Tester*, e por isso ao ser aberto sem *password* será essa a que ele irá tentar usar para desencriptar o ficheiro, e com sucesso.

Mas este tipo de operação não tem grandes vantagens, apesar de ser possível executá-la.

Cuidados a ter quando se encripta um ficheiro de exames

Podem ver acima que ao encriptarmos o ficheiro, nos é dado o nome do ficheiro já encriptado (no exemplo acima "*lab.txt.kt*", e um pequeno alerta de que devemos usar o comando "*history -d -2*" para limpar o comando introduzido, pois se não o fizermos, qualquer pessoa com acesso à máquina onde o exame tenha sido encriptado, pode visualizar a lista de comandos introduzidos, e ver a *password* usada. Com esse comando "*history -d -2*", o comando é apagado, e a *password* desaparece assim e passará a estar guardada apenas na mente de quem criou o exame.

Não se deve encriptar o exame também em máquinas não fidedignas, pois podem conter algum tipo de *key-logger* e assim a *password* ser acedida por terceiros na mesma, ou enviá-las por email ou de outra forma na *Internet* em redes não fidedignas (pode ser vítima de *sniffing* ou outro estratagema), etc.

O poder da encriptação *AES 256* e que *passwords* usar

Parte da força da encriptação reside na palavra-passe usada, e não apenas no algoritmo. Quanto mais caracteres tiver a palavra passe, mais forte fica.

Pois se cada caractere pode ter 256 valores diferentes (2 elevado a 8 bits), poderíamos ter 281.474.976.711.000 combinações de palavras-passes diferentes, ou até um pouco menos se contarmos apenas uns 128 bytes (os caracteres normalmente digitados mais uns extra), o que já dificultaria, se usássemos símbolos à mistura como "$&#)(!/)(" para forçar o uso desse tipo de *brute-force* e tentar invalidar assim ataques por dicionário.

Mas se usarmos uma palavra-passe como "testes", qualquer ataque de dicionário poderia facilmente descobrir a *password* em "pouco" tempo.

Por isso é importante usar *passwords* minimamente grandes, com símbolos diferentes, para evitar ao máximo ataques de dicionário, como por exemplo "oiuW()/"ioYU#98fh#"(hfoiah89"(9h", e isto usando 32 caracteres (para quê mais?).

A pessoa cria a *password* e guarda-a bem, ou então algo fácil de memorizar, como "Este!é"o#exame$da%turma&1975.", que como podemos ver, é fácil de memorizar, tem letras maiúsculas e minúsculas, vários símbolos (os das teclas numéricas com *shift* e seguidos, e até números tem, e seria uma palavra-passe mais do que forte.

Agora, qual o poder máximo de uma encriptação deste género?

Vamos colocar as coisas desta maneira, que é uma maneira engraçada que já vi ser usada antes:

Dizem que existem perto de uns 1.3*10^50 átomos no nosso planeta, na sua totalidade (uns 1.300.000.000.000.000.000.000.000.000.000.000.000.000.000.000.000.000.000 átomos). Se cada um destes átomos fosse um computador, e todos juntos estivessem a tentar 10.000 milhões de possíveis chaves (possíveis *passwords*) por segundo, mesmo assim, ainda levariam 2.840 milhões de anos todos juntos a testar todas as combinações possíveis para desencriptar o ficheiro.

Não é possível quebrar uma *password* super forte encriptada com este algoritmo de encriptação (neste caso o *AES 256*) se a palavra passe for suficientemente boa, a única forma de o fazer, seria por métodos alternativos, seja descobrindo falhas não conhecidas ainda no algoritmo, ou sacando a palavra-passe ao formador/professor de alguma forma.

Agora se alguém usar a *password* "testes", ela poderá ser desencriptada em bastante menos tempo, como é óbvio. Mesmo assim usei uma pequena protecção extra, para dificultar a tarefa de desencriptação, mas não passa do velho método de adicionar fechaduras à porta da

casa, ou seja, pode demover alguns assaltantes de a roubar, mas se algum estiver mesmo motivado, acabará por entrar na mesma.

Por isso, usar palavras-passes o melhores possível, pois um algoritmo de encriptação poderoso de nada serve se quem o utiliza usar *passwords* fracas, é o típico exemplo de uma corrente ser tão forte quanto o seu elo mais fraco.

Relembra-se que a encriptação usada é tão potente no dia de hoje (que posso dizer que vai além do simples algoritmo *AES 256*), que mesmo que usasse o algoritmo *AES 256* sem nada mais, necessitaríamos hoje em dia de milhões de milhões de anos para tentar desencriptar com métodos *brute-force* e tecnologia actual, levaríamos largas vezes muito mais tempo do que a própria idade do Universo.

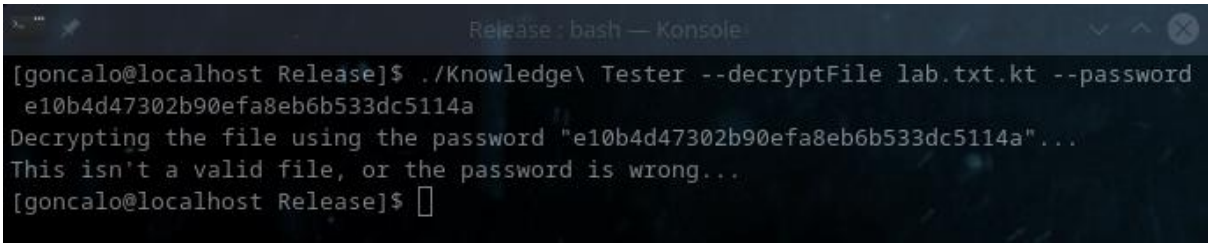
É por isso um método seguro, e um formador ou professor pode literalmente enviar o exame aos seus formandos/alunos, um ano antes, dizendo "está aqui o exame que vão fazer daqui a um ano", e dar-lhes a palavra-passe apenas um ano depois (para abrirem o exame), que por muito que tentem, nunca conseguirão adivinhar a palavra-passe, a não ser que a conseguissem sacar do próprio formador, claro está.

Isto torna o método seguro até para formações à distância.

Como desproteger um exame

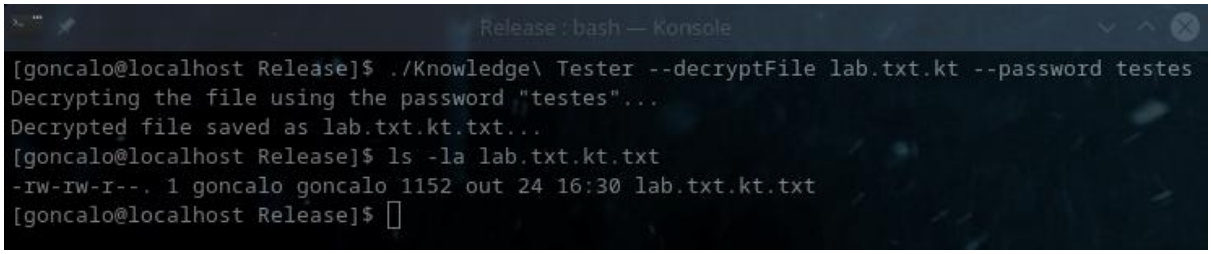
Para desproteger um exame, só necessitamos de ter a *password* usada originalmente para o criar, e usar o parâmetro "*--decryptFile*" seguido do nome do ficheiro, e o "*--password*" seguido da *password* original usada para o encriptar inicialmente.

No exemplo abaixo podemos ver o que poderia ser um formando a tentar desencriptar o exame usando a *password* que é dada para os formandos abrirem o exame, e podem ver que falharia:



Isto porque tal como foi explicado antes, só mesmo com a *password* original, neste caso a "testes", poderá ser desencriptado um exame.

Vamos abaixo ver que com um simples "*Knowledge\ Tester –decryptFile lab.txt.kt –password testes*", o exame é desencriptado, e é gerado um novo ficheiro com o conteúdo desencriptado, adicionando-se um "*.txt*" à extensão original:

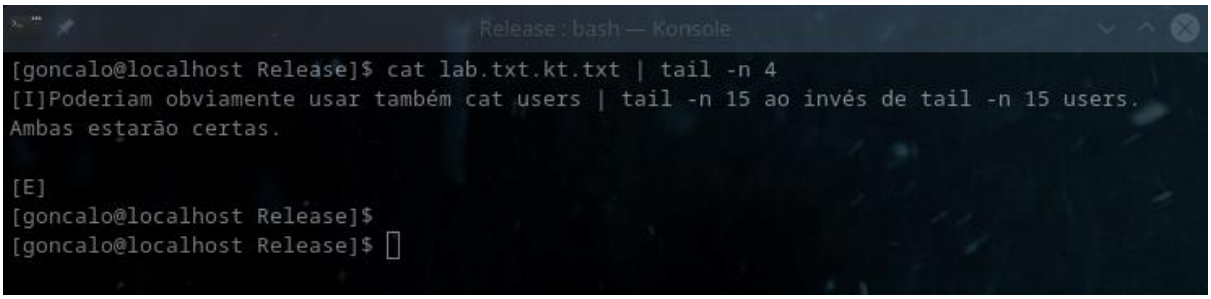


Como podem ver pela imagem acima, o ficheiro "*lab.txt.kt.txt*" foi criado, e contém o conteúdo desencriptado do exame original, o que pode ser útil caso o formador tenha perdido o exame original em texto e necessite de o recuperar.

Pode é ser algo confuso o sistema adoptado, de adicionar "*.txt*" quando desencriptamos, ao nome do ficheiro, e depois "*.kt*" quando o encriptamos novamente, e podemos acabar um dia com um ficheiro de nome "*lab.txt.kt.txt.kt.txt.kt.txt.kt*" se o fizermos de forma indefinida, mas isso só acontece se a pessoa o quiser, pois o formador tem apenas de mudar o nome para o desejado com um simples "*mv lab.txt.kt.txt lab.txt*" e tudo fica perfeito.

Lembrem-se que depois de desencriptado, o ficheiro continuará a não ter qualquer referência à *password* original, ela nunca é escrita nem contida em lado nenhum.

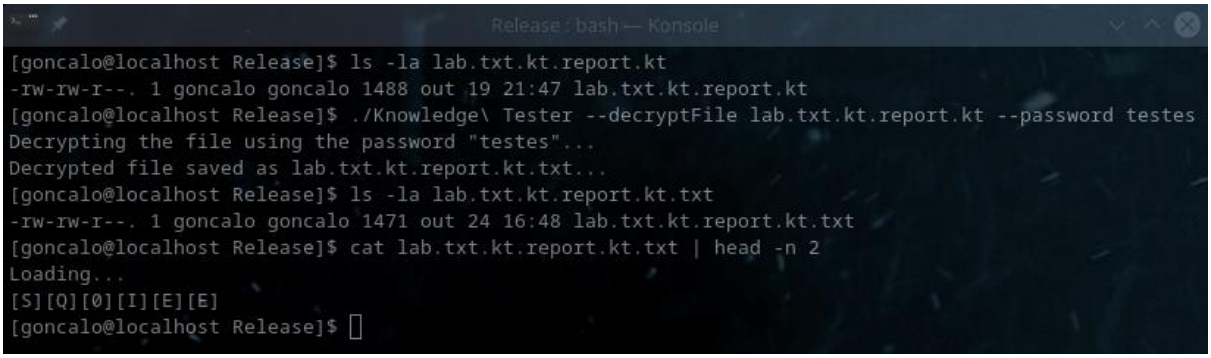
Podemos ver abaixo o conteúdo do ficheiro de exame já desencriptado, para verem que está em texto legível já:



Como visualizar o conteúdo de um relatório encriptado

Com os relatórios encriptados, o método é o mesmo do usado para desencriptar exames encriptados, ou seja, corremos o "*Knowledge\ Tester*" seguido de "*--decryptFile*" e nome do ficheiro, e um "*--password*" seguido da *password* usada para criar o exame original, só que ao invés de estarmos a desencriptar um exame, estaremos a desencriptar um relatório gerado por um exame, mas o funcionamento por detrás é o mesmo, são desencriptações aos ficheiros.

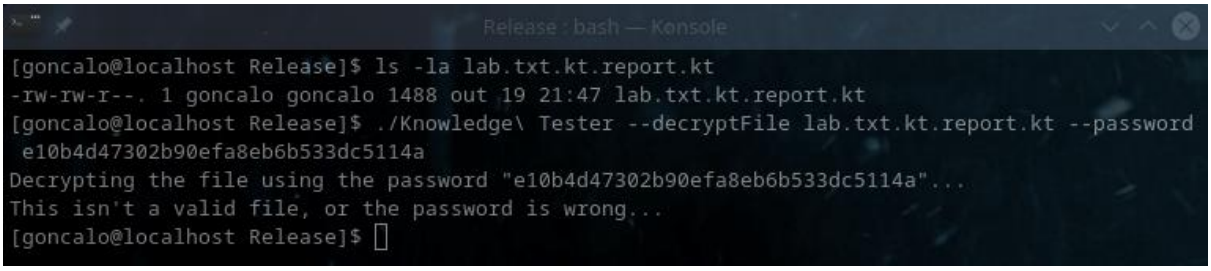
Abaixo fica um exemplo da desencriptação de um relatório gerado por um exame:



Como podem ver pela imagem acima, o ficheiro "*lab.txt.kt.report.kt*" foi desencriptado (nome confuso não é?), e o seu conteúdo passou a estar em modo de texto "*raw*", legível.

Depois cabe ao formador mudar o nome para o desejado.

Escusado será dizer que a *password* dada aos alunos para desencriptar, a "*e10b4d47302b90efa8eb6b533dc5114a*", não resultaria, pois assim qualquer aluno poderia desencriptar o relatório e alterá-lo sem problemas...



Como podem ver acima, a *password* dada aos estudantes, não desencripta o relatório, pelo que ao ser usada recebemos um erro de "*This isn't a valid file, or the password is wrong...*", e neste caso sabemos que é a *password* que está errada, pois o ficheiro é mais do que válido e já foi desencriptado antes com a *password* correcta.

Como forçar opções num exame

Podemos forçar opçõesn um exame, fazendo com que estejam já embutidas no mesmo, e evitando que sejam alteradas à mão na linha de comandos, com o argumento "*--forceArguments*" seguido do nome do ficheiro.

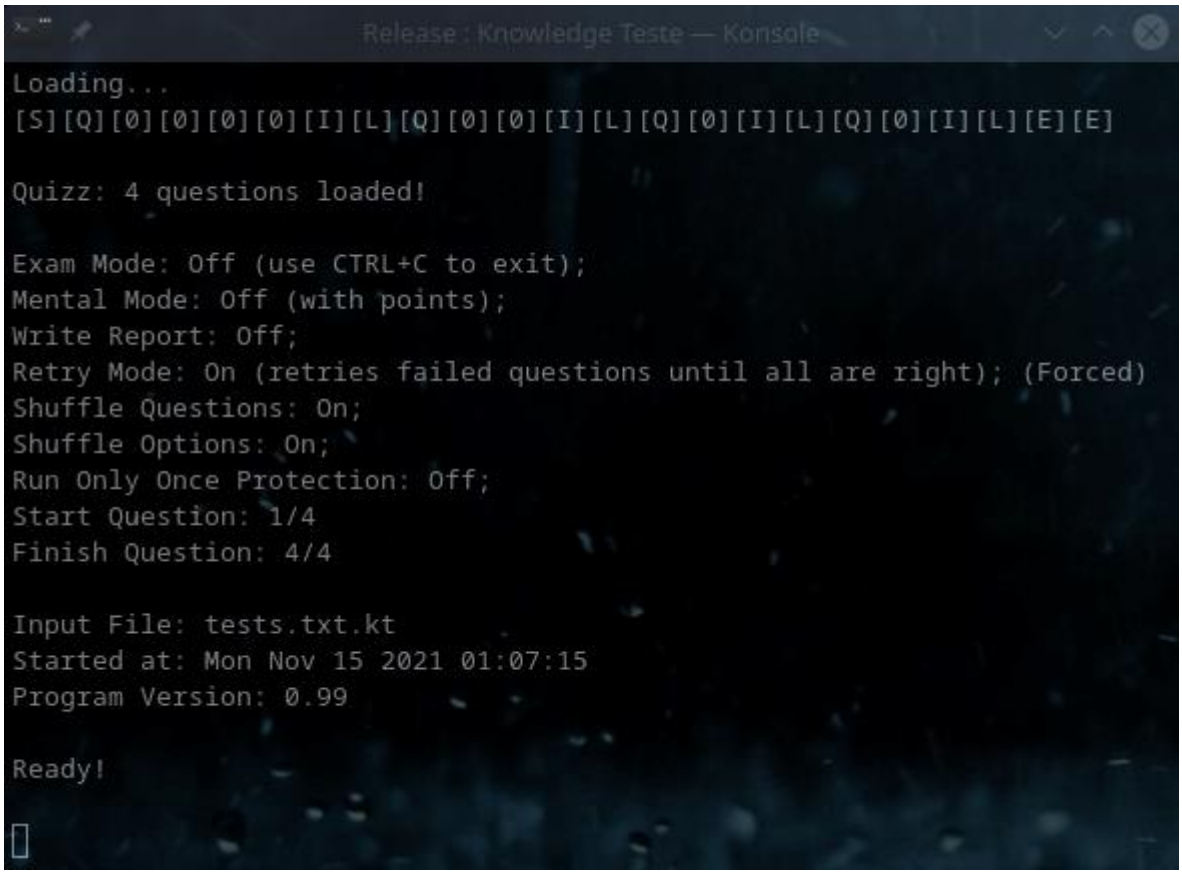
Por exemplo, se adicionarmos o argumento "*--forceArguments*" seguido do nome do ficheiro, e colocarmos o argumento ou argumentos em questão, esse ficheiro será encriptado e com as opções já embutidas, sem permitir que o utilizador final as altere.

Como exemplo, podemos tentar um "*Knowledge\ Tester --forceArguments ficheiro.txt --retryOff --password hello*", para forçar o argumento "*--retryOff*" como estando sempre activado, e mesmo que tentemos arrancar o exame com um argumento "*--retryOff 0*" (para o desligar), ele estará sempre ligado na mesma.

É importante definir a *password* para que só através dessa *password* original consigamos desproteger o exame e remover a opção.

É importante reparar que num "*--retryOff*" ou "*--shuffleOptionsOff*", que por defeito desligam essas funcionalidades, podemos adicionar um "*0*" à frente, e assim elas têm o efeito contrário, ligando as mesmas.

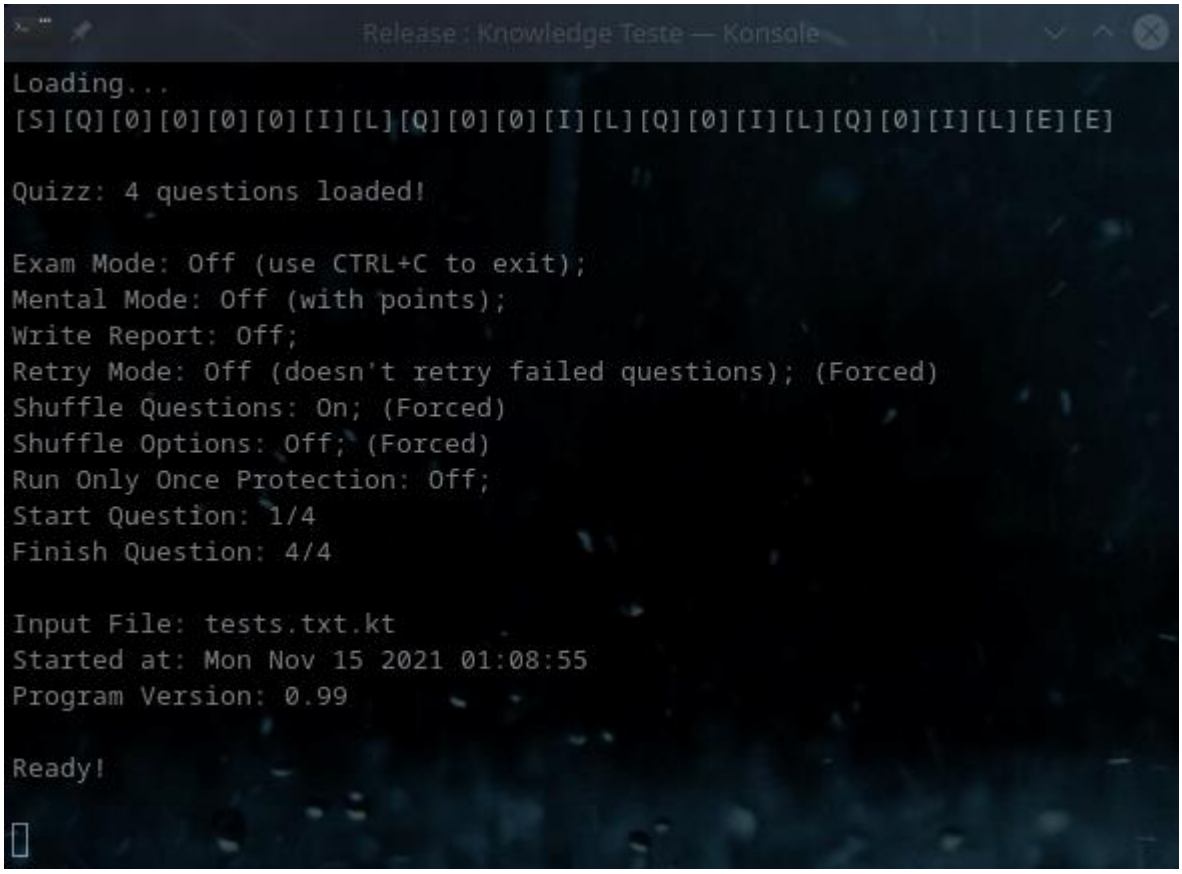
Assim, por defeito elas têm um valor de "*1*", mas podemos forçar um valor negativo, como "*0*":



No exemplo acima, o "*Retry Mode*", definido pelo argumento "*--retryOff*", está definido como sendo "*0*", ou seja, o modo de "*Retry*" está definido como estando sempre ligado, daí ter um "*(Forced)*" à frente no ecrã de entrada inicial.

Se não tivessemos usado o "*0*" à frente do argumento "*--retryOff*", teríamos o "*--retryOff*" como sendo verdadeiro, e aí nunca estaria ligado o "*Retry Mode*", nem que o tentássemos ligar à mão.

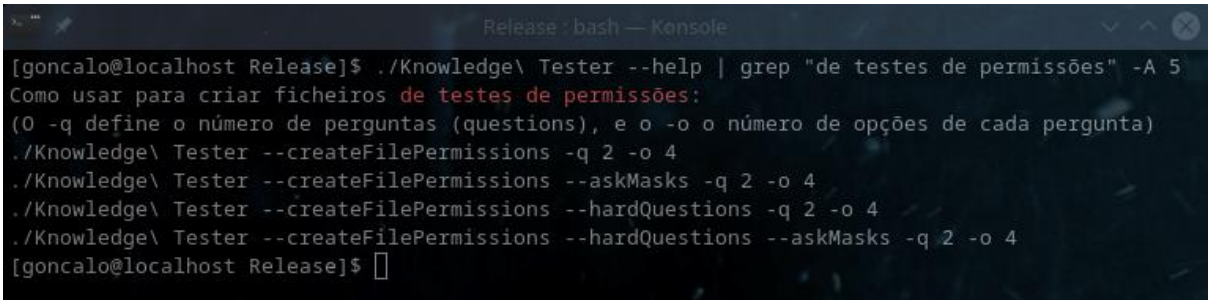
No exemplo abaixo, temos esse modo invertido, sempre ligado o "*--retryOff*", e com isso, sempre desligado o "*Retry Mode*", e ainda forçámos alguns outros parâmetros, com a linha de comandos "*./Knowledge\ Tester --forceArguments tests.txt --retryOff 1 --password hello --shuffleOptionsOff 1 --shuffleQuestionsOff 0":*



Podemos ver acima assim vários parâmetros forçados (com um "*(Forced)*" à sua frente), sendo que esses parâmetros não podem ser alterados através da linha de comandos, pelo que só a pessoa com a *password* original os poderá alterar, o que é muito útil em caso de exames em que desejamos que os formandos não consigam alterar certas opções.

Ajudas e Informações

Se escreverem na linha de comandos "*Knowledge\ Tester –help*", ser-vos-ão mostrados os vários parâmetros e como os usar, apesar de não ser algo tão completo como este manual:



Opções Disponíveis

Repetição de perguntas falhadas

Repetição de perguntas falhadas em exercícios

Com a opção "*--retryOff*" desligamos uma opção que existe activada por defeito, que é a de repetir toda as perguntas falhadas.

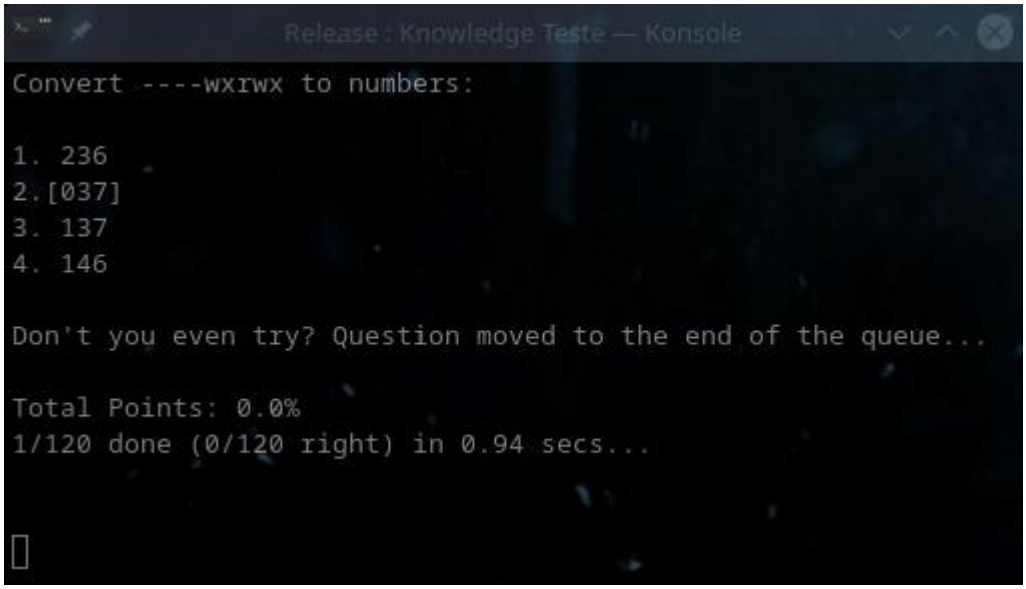
Para facilitar a aprendizagem, e memorização de comandos e matéria, é dada por defeito ao aluno que pratica um exercício, a chance de repetir as perguntas falhadas no fim do mesmo.

Assim, se o aluno falhar a resposta, essa pergunta volta para o fim da lista, e chegando ao fim do número de questões que o exame tem, se existirem questões nesse fim da lista, elas serão apresentadas de novo ao aluno, e se ele continuar a falhar indefinidamente, as perguntas continuarão a lhe ser apresentadas, até que finalmente ele as responda correctamente.

Isto reforça a aprendizagem, porque a pessoa tem de repetir o que não sabe, até memorizar/aprender a matéria, pois o exercício só termina após as perguntas serem todas respondidas correctamente.

Desta forma, se voltar a começar o exercício do zero, ele terá muito mais probabilidades de o terminar sem falhas logo à primeira vez.

Abaixo vemos um exemplo de uma pergunta a ser movida para o final da lista:



Podem ver pela mensagem "*Question moved to the end of the queue...*" acima, que a mensagem foi movida para o fim da lista, sendo que o "*Don't you even try?*" está lá por eu não ter respondido a nada, tendo simplesmente pressionado a tecla *Enter* para seguir em frente.

Se iniciarmos o exercício com o parâmetro "*--retryOff*", elas não serão movidas para o fim do exame, sendo que o utilizador se as falhar ou não responder, não voltará a ser questionado com as mesmas no mesmo exercício, nem sequer no fim da lista.

Repetição de perguntas não respondidas em exames

Durante a execução de exames, também existe esta funcionalidade, mas funciona de forma diferente.

Aqui, não são as perguntas falhadas que passam para o fim do exercício, mas sim as perguntas não respondidas, senão o aluno teria sempre 100% pois responderia a tudo até saber a matéria.

Neste caso, o aluno se responder mal a uma pergunta, é contabilizado com uma nota parcial, ou até 0% se o falhar foi total, e a pergunta não volta a aparecer.

Mas se o aluno pressionar apenas a tecla *Enter*, a pergunta será movida para o fim da lista, e será questionado com a mesma ao chegar ao fim do exame, sendo que o contador de perguntas feitas, não será incrementado, para que só quando responder (nem que seja só com uma letra) a todas as perguntas, é que o contador chegará ao fim e o exame terminará.

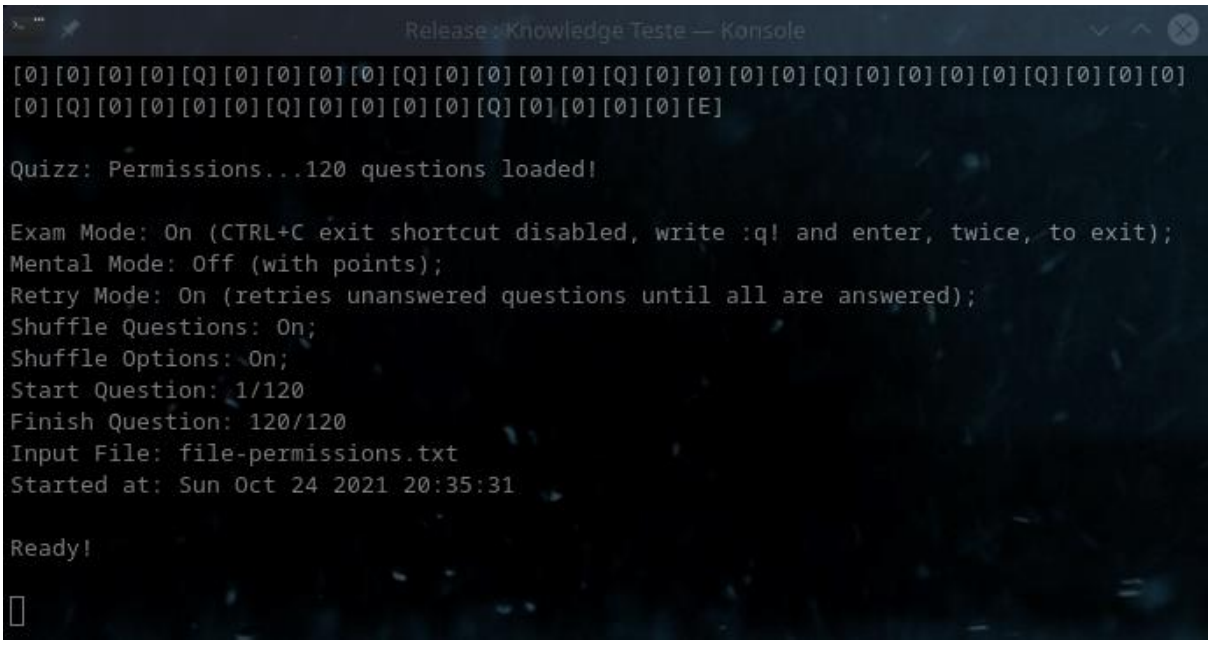
Esta funcionalidade existe para que os alunos nos exames se foquem em responder às perguntas que sabem primeiro, e deixem as que não se lembram bem, para o fim, para não perderem tempo precioso que pode ser gasto a responder às perguntas que sabem.

As perguntas podem ser movidas para o fim da lista as vezes que forem precisas, para que o aluno deixe para o fim só mesmo as que possa não saber mesmo nada.

Neste caso, também o parâmetro "*--retryOff*" funciona, e se for usado, e se um aluno pressionar a tecla *Enter* numa pergunta de um exame, ela não seria movida para o fim da lista, mas simplesmente contabilizada como falhada e com 0% de nota.

O aluno examinado pode saber se a opção em causa está ligada ou não, para saber se pode pressionar a tecla *Enter* para deixar perguntas difíceis para o fim, ou não, no ecrã inicial de arranque do exame.

Por isso há que estar atento ao ecrã de arranque do exame, e verificar se a opção "*Retry Mode*" está activa ou não, e o que diz:



E como podem ver acima, após o "*Retry Mode*", temos escrito "*On*" e "*(retries unanswered questions until all are answered)*", ou seja, o aluno poderá mover perguntas não respondidas para o fim da lista sem problemas.

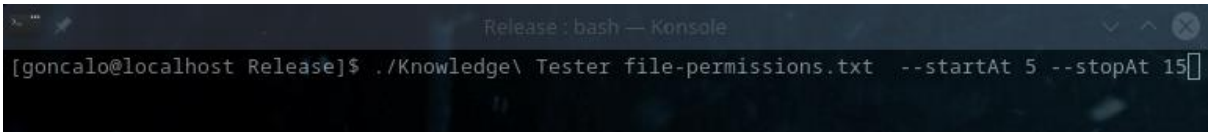
Correr um intervalo de perguntas apenas

É importantíssimo saber que é possível correr apenas um intervalo de questões num dado ficheiro de perguntas.

Assim, se tivermos um ficheiro de exercícios com umas 100 perguntas, não será fácil aprender as respostas, mesmo que as falhadas sejam enviadas para o fim, com 100 perguntas, pois se falharmos a 3ª e virmos a resposta correcta, não vamos recordar a mesma tão bem, 97 perguntas depois.

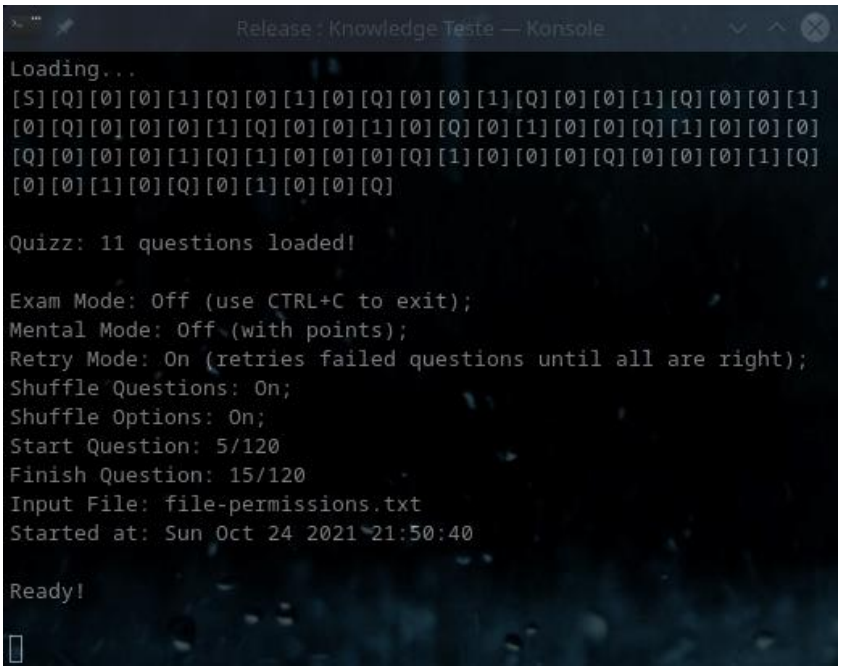
Nesses casos, quando os ficheiros de perguntas são grandes, o formando ou aluno, pode usar as opções "*--startAt*" e "*--stopAt*" para definir em que pergunta começa e acaba o exercício.

No exemplo abaixo, definimos que num exame de 120 perguntas, vamos executar apenas as perguntas entre a quinta e décima quinta perguntas:

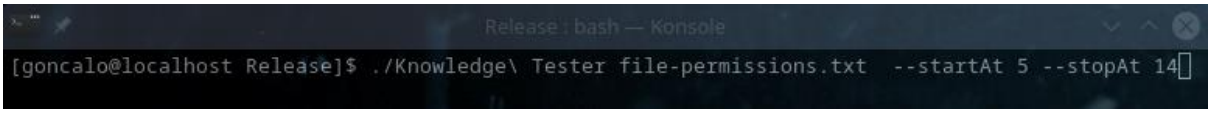


Na opção acima, com o "*--startAt 5*" e o "*--stopAt 15*", vamos correr um exame onde só responderemos às questões entre a quinta e a décima quinta, ignorando todas as outras, e apesar de serem 120 perguntas, como só vamos responder a 11, cada uma terá um valor de 9,09% de nota, obviamente.

E porquê 11 perguntas? São 11 porque se definimos começar em 5 e acabar na pergunta 15, ele vai correr as 10 entre 5 e 14, e a 15ª também. Ou seja, as perguntas definidas no argumento do parâmetro "*--startAt*", e as definidas no argumento do parâmetro "*--stopAt*", são todas contabilizadas:



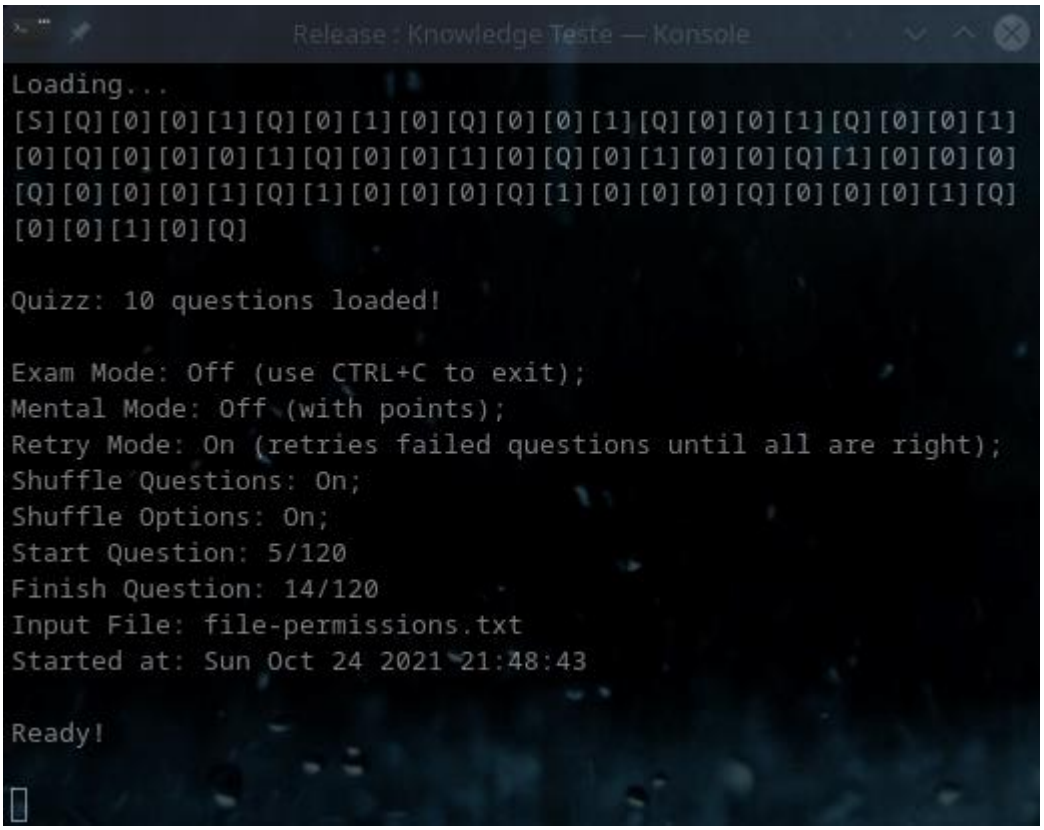
Assim, se desejarmos apenas 10 perguntas, entre a 5ª e a 15ª, excluíndo a 15ª, ou seja, na realidade, responder às perguntas 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, mas ignorando a 15ª, teríamos de correr o comando com o "*--startAt*" em 5 e o "*--stopAt*" em 14:



Pelo exemplo acima, já teríamos 10 perguntas, de 4 a 14, e valendo cada uma 10% de nota, por serem dez no tota, e isto pode parecer confuso à partida, mas não é, pois tem mais sentido dizer para se correr da pergunta 1 à 10, 11 à 20, 21 à 30, 31 à 40, do que dizer que arrancamos com 1 à 11, 11 à 21, 21 à 31, etc.

Faz mais sentido assim, incluir ambas as perguntas, a do começo e a do fim.

Vejamos o resultado do comando acima:



E assim conseguem-se filtrar algumas perguntas num exame com muitas perguntas, e dividindo-o assim em várias secções, para serem memorizadas facilmente.

É importante referir que mesmo que sejam baralhadas as perguntas, só são lidas no exemplo acima as perguntas da 4ª à 14ª, e só depois de lidas é que são baralhadas, pelo que é seguro filtrar perguntas assim.

Baralhar perguntas e opções

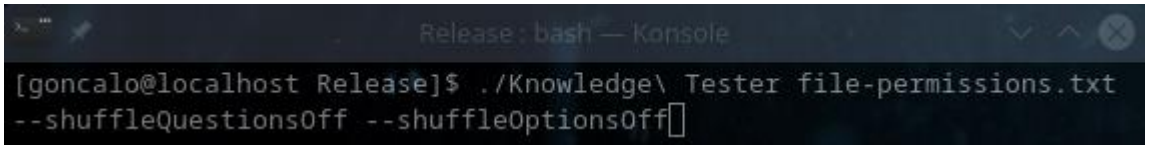
Existem dois parâmetros que nos permitem desligar o que está activo por defeito, que é o baralhar das perguntas e opções das perguntas.

Com o parâmetro "*--shuffleQuestionsOff*" podemos desligar o baralhar das perguntas, e elas aparecerão na ordem em que estão escritas no ficheiro de perguntas ou exame.

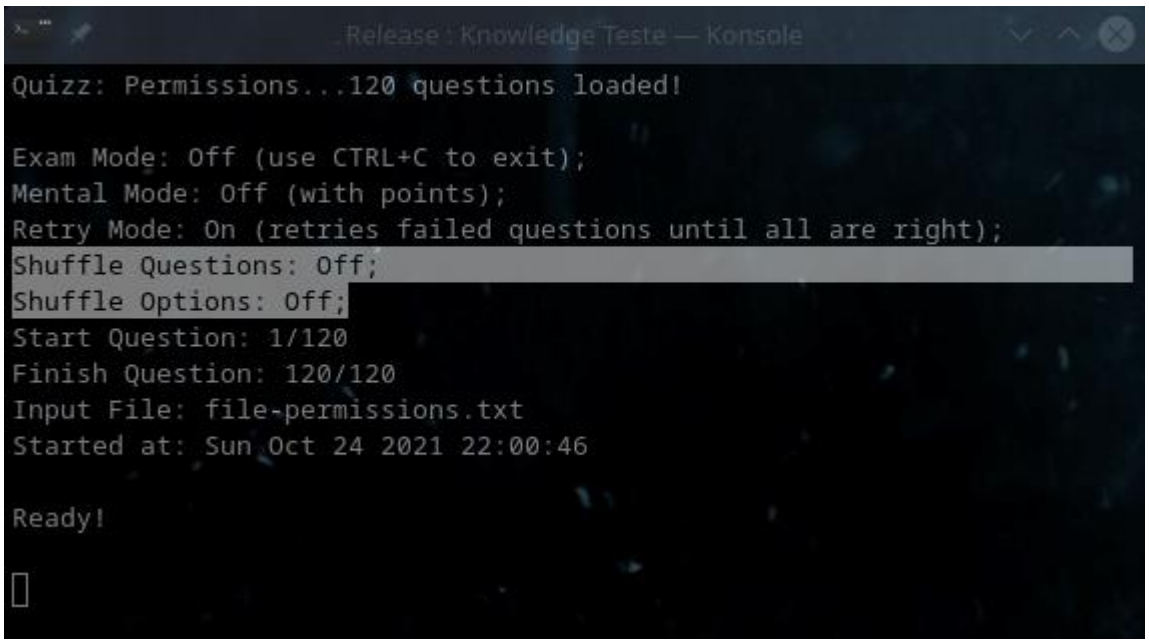
Da mesma maneira, com o parâmetro "*--shuffleOptionsOff*", podemos desligar o baralhar das opções de cada pergunta, pelo que surgirão pela mesma ordem em que estão escritas no ficheiro de perguntas ou exame.

No caso das opções não há muito sentido, mas o desligar o baralhar de perguntas pode ser útil em casos de laboratórios, em que simulamos os passos necessários para uma operação com muitos comandos, em que dedicamos uma pergunta e explicação a cada um dos passos, e em que nesses casos necessitamos de ter perguntas a sair pela ordem em que estão nos ficheiros.

Abaixo fica um exemplo de como se corre um exame com ambas as opções desligadas:



E depois pelo ecrã de arranque veremos que já estão desligados ambos os baralhar de perguntas:



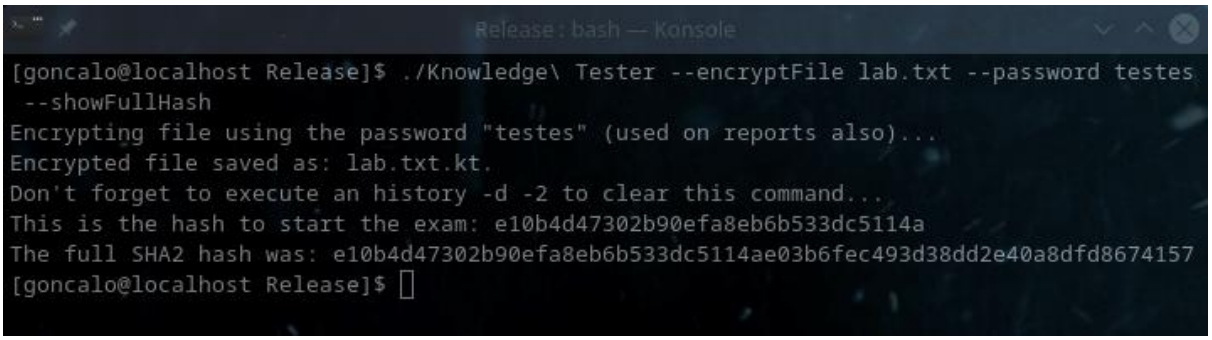
E assim se desligam as funcionalidades de baralhar tanto as perguntas como opções, que podem ser obviamente ambas chamadas individualmente, pois nenhuma delas necessita da outra.

Opções de encriptação de ficheiros

Aqui são apenas referidas algumas poucas funcionalidades, como a de mostrar a *hash* inteira aquando da encriptação de um ficheiro de exame, que por norma ela é encurtada, para facilitar a sua transmissão aos alunos, e por sabermos que mesmo assim não deixa de ser mais do que segura.

A opção *--showFullHash*

Com o parâmetro "*--showFullHash*", surge-nos a linha no fim, que nos diz "*The full SHA2 hash was:*" seguida de uma *hash* inteira:

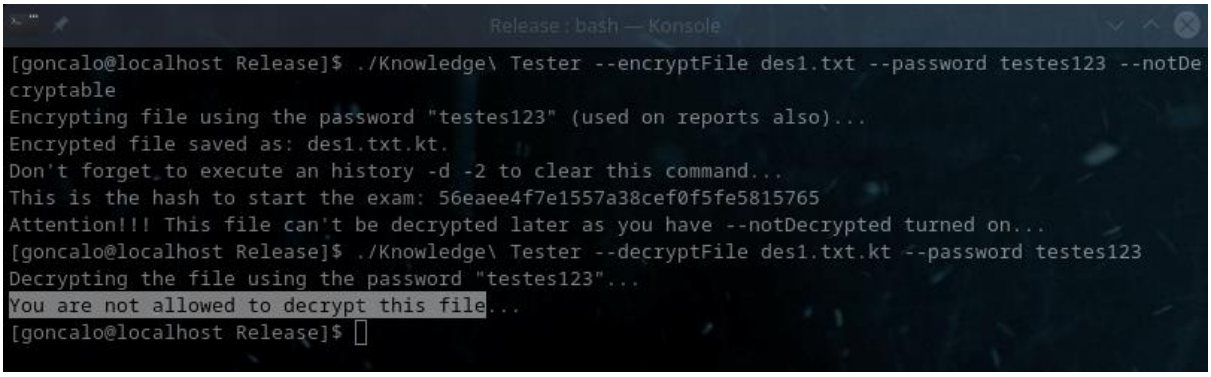


Mas é importante referir que nos basta a *hash* encurtada para abrir exames, tudo o resto é descartado pela aplicação, pelo que este parâmetro só é mencionado aqui para efeitos de teste por parte dos formadores ou outras pessoas que queiram criar exames.

A opção *--notDecryptable*

Com esta opção activada durante uma encriptação de um ficheiro, o mesmo não voltará a poder ser desencriptado, por isso há que ter cuidado com ela, pois se perdemos o ficheiro original, e o encriptado tiver sido criado com esta opção, nunca mais recuperaremos o ficheiro de texto original, pelo que teremos de o recriar do zero se o quisermos alterar.

Abaixo podemos ver como a usar, e como é impossível depois desencriptar o ficheiro mesmo possuindo a *password* original:



O parâmetro de versão mínima ("*--minVersion*")

Este parâmetro pode ser definido em conjunto com o "*--forceArguments*", e deixa-nos registar no exame a informação de que o exame em questão só pode correr num *software Knowledge Tester* com uma versão igual ou acima da definida por este parâmetro.

Essa versão tem de ser dividida por 100.

Se usarmos o parâmetro "*--minVersion 19*", significa que o software só correrá se a versão do *software* em si for *0,19* ou superior, e uma "*99*" significaria versão *0,99*, uma "*1*" significaria versão 1,00, etc.

Isto permite-nos garantir que o exame é corrido numa aplicação que tenha todas as funcionalidades requeridas pelo exame em questão, para evitar que os resultados do mesmo possam ser manipulados, ou o exame executado de maneira não inicialmente pensada.

Para isso só temos de colocar esta linha à frente da linha de comandos que corramos com a opção "*--forceArguments*", que deverão procurar no índice deste manual, e ela será guardada no exame.

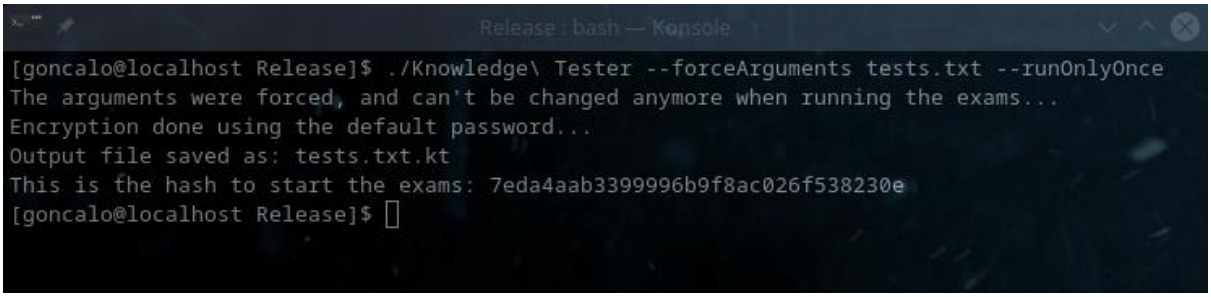
É de notar, que só é guardada em exames encriptados, pelo que se desencriptarmos o exame, ela deixará de estar activa (ver mais nas secções sobre como encriptar exames).

Proteger exames para só executarem uma vez ("*--runOnlyOnce*")

Com o parâmetro "*--runOnlyOnce*" inserido com o "*--forceArguments*", o exame passa a estar protegido e só deixa que seja corrido uma única vez, e quem tentar correr novamente verá a sua tentativa frustrada.

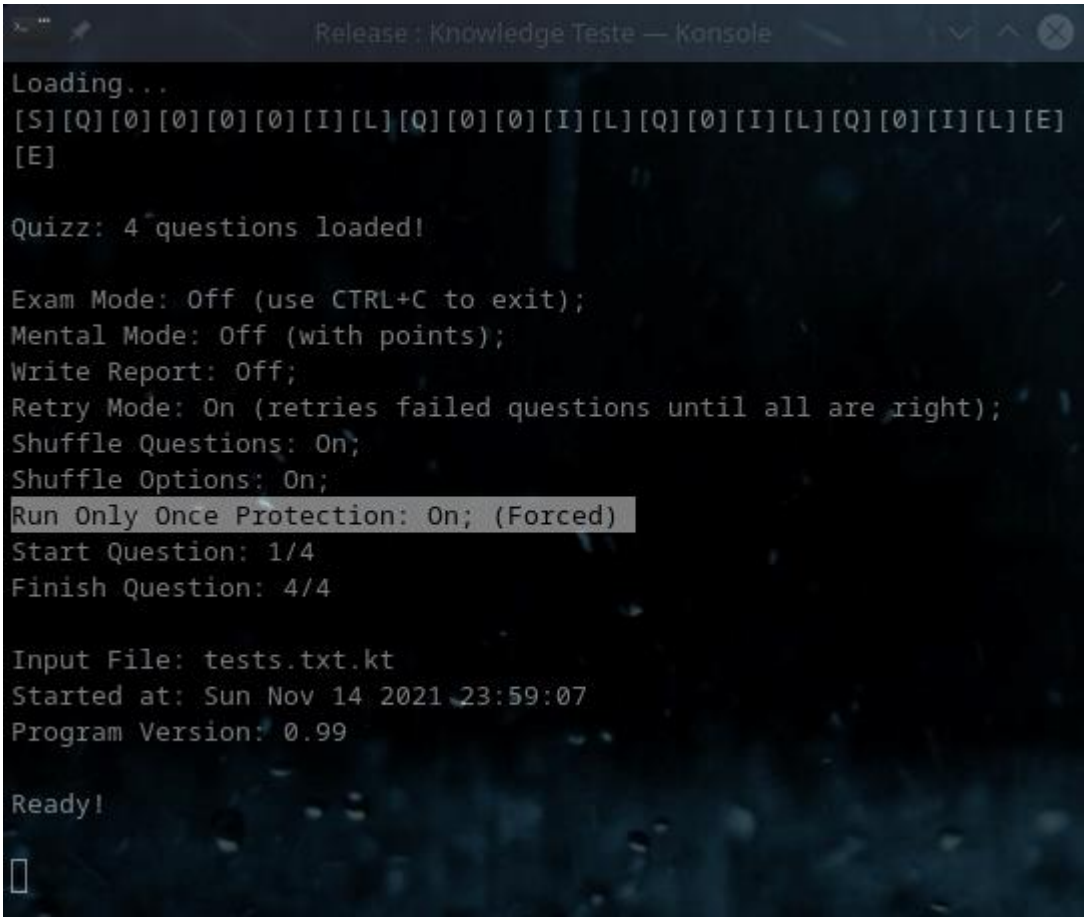
Esta protecção existe para evitar que alunos cancelem o seu exame assim que começa a correr mal e o recomecem, e com ela, assim que o exame é iniciado, não pode voltar a ser reiniciado.

Quando usada junto com o argumento "*--forceArguments*", ela é guardada no ficheiro na sua forma encriptada, e só desaparece se usarmos a opção contrária do "*--removeRunOnlyOnceProtections*".



Quando forçamos argumentos, como o ficheiro é encriptado por defeito, é-nos dada uma *password* para que possa ser aberto, e por isso se queremos que ele veja as opções forçadas com uma determinada *password* que não a por defeito, devemos acrescentar o argumento "*--password* palavra-passe" à linha de comandos.

Mas se tentarmos executar o comando, veremos que já tem forçada a opção de correr apenas uma vez, aparecendo "*(forced)*", à frente da mesma:



Se tentarmos correr o mesmo exame novamente, teremos nada mais do que uma mensagem de erro dizendo que tal já não é possível porque o exame já correu uma vez:

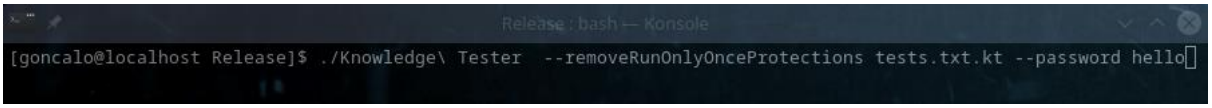
E é de notar que nem que se troque o exame por um exactamente igual, se consegue executar tal exame novamente, é uma protecção básica e simples, mas minimamente eficaz, que é o que existe de momento nesta fase de desenvolvimento do programa.

Remover protecções de correr apenas uma vez ("*--runOnlyOnce*")

Com o argumento "*--removeRunOnlyOnceProtections*", usado em conjugação com o argumento "*--forceArguments*", o ficheiro encriptado é aberto, essa sua opção é removida, e é guardado já sem essa protecção.

Este argumento existe para o caso de alguém querer remover essa protecção a um exame, para que os alunos o possam correr as vezes que forem necessárias.

Para isso temos de invocar esse argumento "*--removeReadOnlyProtections*" seguido do nome do ficheiro em questão, sem esquecer de referir a *password* original usada para encriptar/proteger o ficheiro, e tem de ser essa original, senão qualquer aluno/formando seria capaz de o fazer com a *password* de abertura dos exames:



Após remover as protecções, teremos a seguinte mensagem, e poderemos correr novamente o exame agora desprotegido:

Geradores de Exercícios Embebidos

Existem vários exercícios já pré-embutidos no *software*, como um de mover pastas, um de permissões de ficheiros, e um de mover pastas e mudar nomes.

São exercícios que compensa existirem em separado, pois fazem o aluno ou formando praticar em poucos minutos algo que levaria horas de uso real do sistema a praticar, em

termos de experiência, pois no dia-a-dia é raro alguém estar a mudar constantemente permissões de ficheiros, e com este exercício acabam por o fazer centenas de vezes em dez minutos.

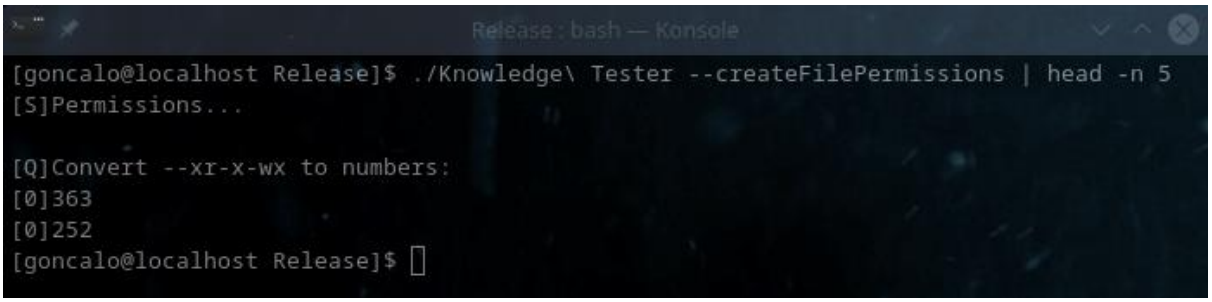
Este tipo de exercícios compensa ter neste tipo de *software*, e é provável que outros sejam adicionados em futuras versões deste *software* gratuito.

Exercícios de Permissões de Ficheiros

Como criar o exercício

O exercício é gerado através de uma linha de comandos como tudo o resto, com o comando e parâmetro "*Knowledge\ Tester --createFilePermissions*", e tem quatro estilos possíveis de exame, conseguidos com a combinação de duas opções, a "*--askMasks*", e a "*--hardQuestions*", que conjugadas nos dão quatro tipos de exercícios possíveis.

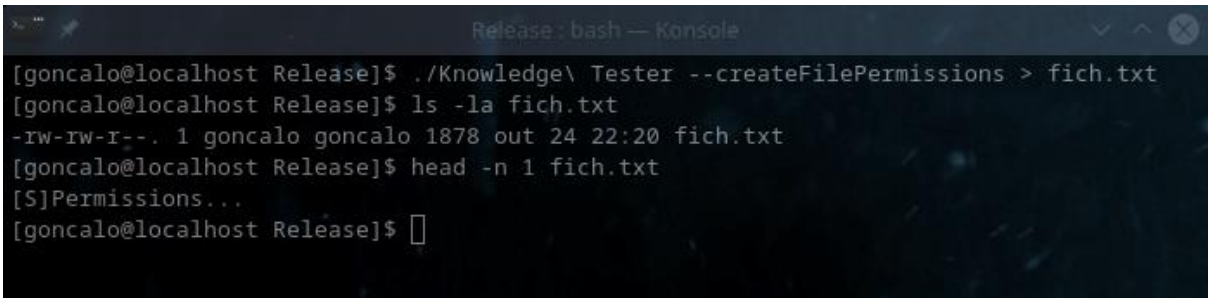
Abaixo temos o invocar do comando que nos dá as questões:



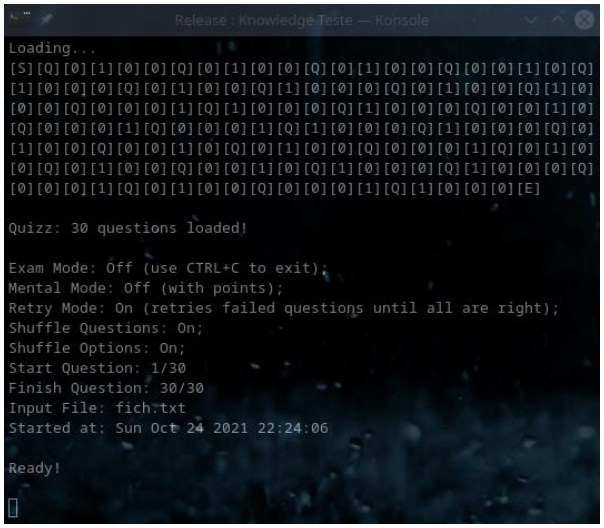
Mas se como podemos ver acima (filtrado o *output* com o comando "*head -n 5*" para mostrar apenas as primeiras 5 linhas do exercício), o *software* envia-nos as perguntas para o ecrã.

Sempre que repetirmos este comando, poderemos ver que nos cria questões diferentes, "aleatoriamente".

Para fazermos exercícios com este resultado, temos de acrescentar um "*> ficheiro.txt*" ao fim da linha, para gerar o ficheiro com respostas:



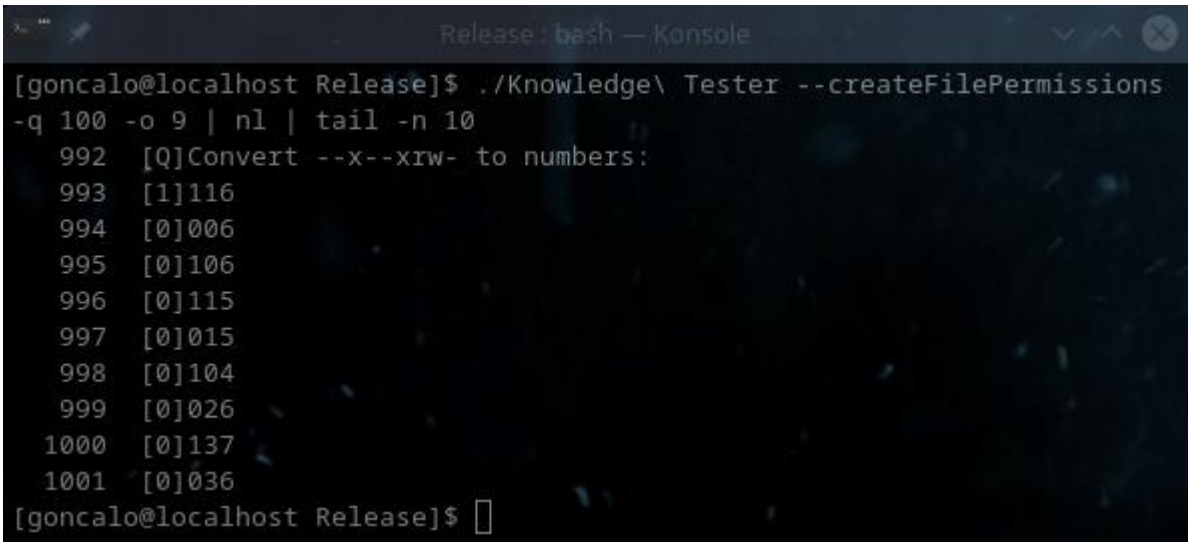
Na imagem acima, vemos que o ficheiro de nome "*fich.txt*" foi assim gerado, e que contém neste momento as perguntas para serem respondidas, que são por defeito 30:



Como definir o número de questões e opções por exercício

Como podemos alterar o número de questões a ser apresentada por defeito?

Se executarmos "*Knowledge\ Tester --help*" podemos ver que temos os parâmetros "*-q*" e o "*-o*", sendo que o "*-q*" nos permite definir o número de questões que vamos ter no exercício, e o "*-o*" nos define o número de opções por cada questão do exercício:



No exemplo acima podemos ver que definimos 9 opções por questão, e 100 questões (das quais só vemos uma no ecrã graças ao comando "*tail -n 10*"), e podemos ver que foram 100 questões, dado que com o comando "*nl*" (que nos coloca o número da linha antes de cada

linha), podemos ver que são perto de 1000 linhas, que correspondem a 100 questões onde cada questão tem uma linha para a pergunta e nove para as opções.

É importante referir que o número mínimo de opções é 2, sendo que se for definido um número inferior, teremos 2 opções por questão.

O número mínimo de questões é 1, sendo que se definirmos 0 questões, teremos uma.

E o número máximo de opções é 9, sendo que se definirmos mais do que 9, teremos 9, e isto é importante, dado que só podemos responder usando as teclas de 1 a 9, e se tivéssemos 20 opções e a certa fosse a opção 15, não conseguiríamos responder com o premir de uma tecla, daí são limitadas a 9, para serem respondidas com as teclas de 1 a 9.

Opções disponíveis

O parâmetro --askMasks

Este parâmetro neste exercício faz com que sejam pedidas ao utilizador as máscaras "*rwxrwxrwx*" sendo dados os números (ao invés do contrário):

Este tipo de perguntas em conjugação com as anteriores enriquecem o exercício, e será explicado mais abaixo como as juntar num único.

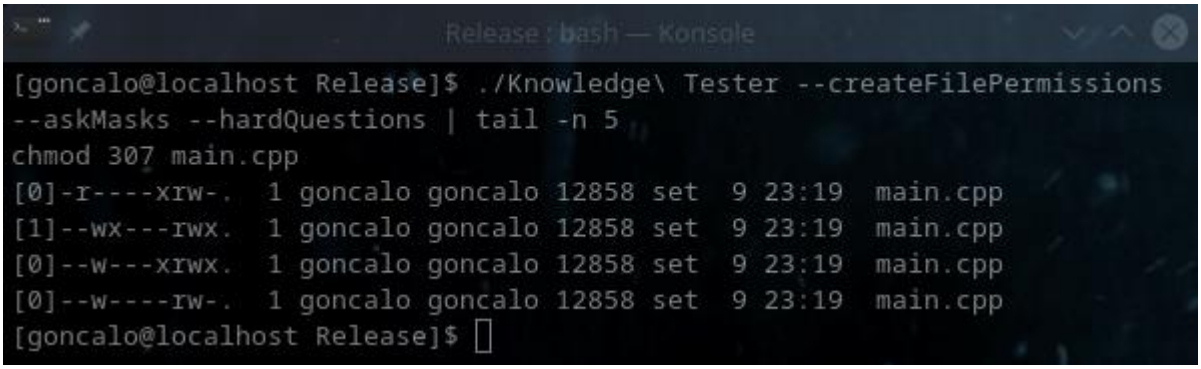
O parâmetro --hardQuestions

Com este argumento, o "*--hardQuestions*", tudo é ligeiramente dificultado, pois as máscaras "*rwxrwxrwx*" passam a ser iguais ao que se vê numa *shell* tradicional de *Linux*, tendo um "-" antes de cada linha (a representar o ficheiro), e podemos ver como fica aqui no método tradicional sem "*--askMasks*" mas com *"--hardQuestions*":

Podemos ver acima que a pergunta já fica mais realista, e algo mais difícil, daí a *tag* ter como nome "*--hardQuestions*", apesar de que na realidade foi também por falta de imaginação da minha parte para um nome melhor.

Por norma o ideal é sempre ser usada a *tag* "*--hardQuestions*", pois é mais realista, e o objectivo é sempre o aluno/formando estar preparado o mais possível para a realidade, para distribuições reais de *Linux* com as suas *shells* reais.

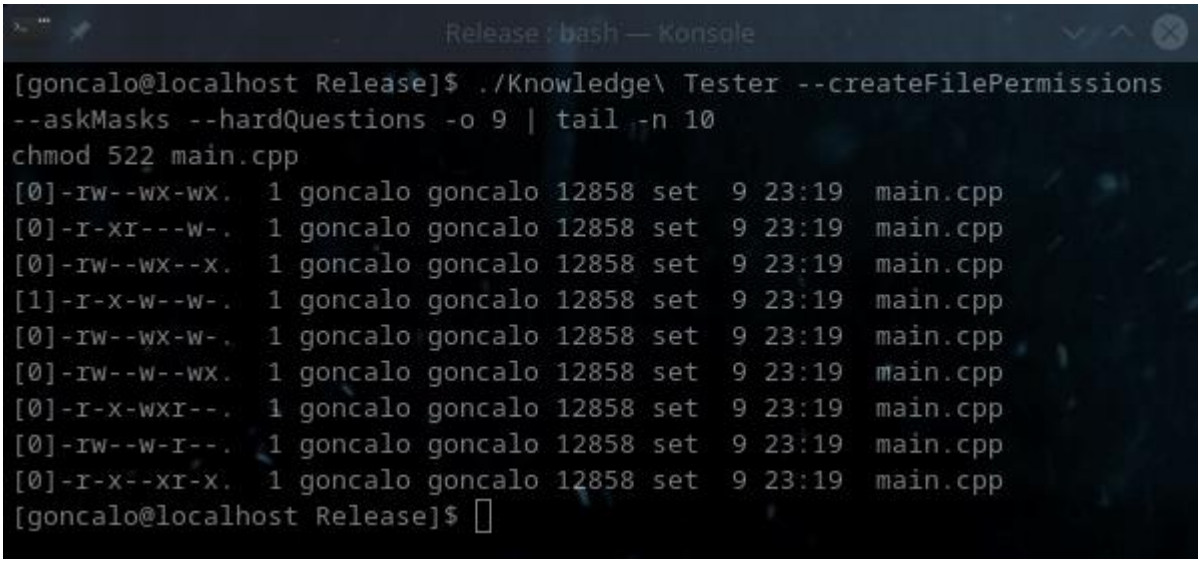
Vejamos como fica a mesma *tag* "*--hardQuestions*" conjugada com a *tag* "*--askMasks*":



Bastante mais realista, e por norma basta que o formando ignore o "-" inicial, que é usado para no *Linux* representar um ficheiro (sendo que "*d*" é para directorias, "*l*" para *links*, etc), e tudo se torna mais fácil.

Assim, o ideal é conjugar exercícios com "*--askMasks*", com exercícios com "*--hardQuestions*", e outros com a segunda mas sem a primeira, e veremos mais abaixo como os criar.

Agora para dificultar mesmo a tarefa, basta juntarmos o parâmetro "*-o 9*" para aumentar o número de opções, e ficaremos com:



Até dá vontade de tentar, não?

Na realidade, apesar de esta pergunta ser uma brincadeira considerada por alguns de mau gosto, isto torna-se bem mais fácil do que parece, com alguma prática, pelo que o ideal é a pessoa praticar sempre com o número máximo de opções (9) e com as *tags* "*--hardQuestions*", tenham a "*--askMasks*" activa ou não.

Juntar vários exercícios num só

Agora, como juntar de forma fácil os quatro tipos de exercícios num só?

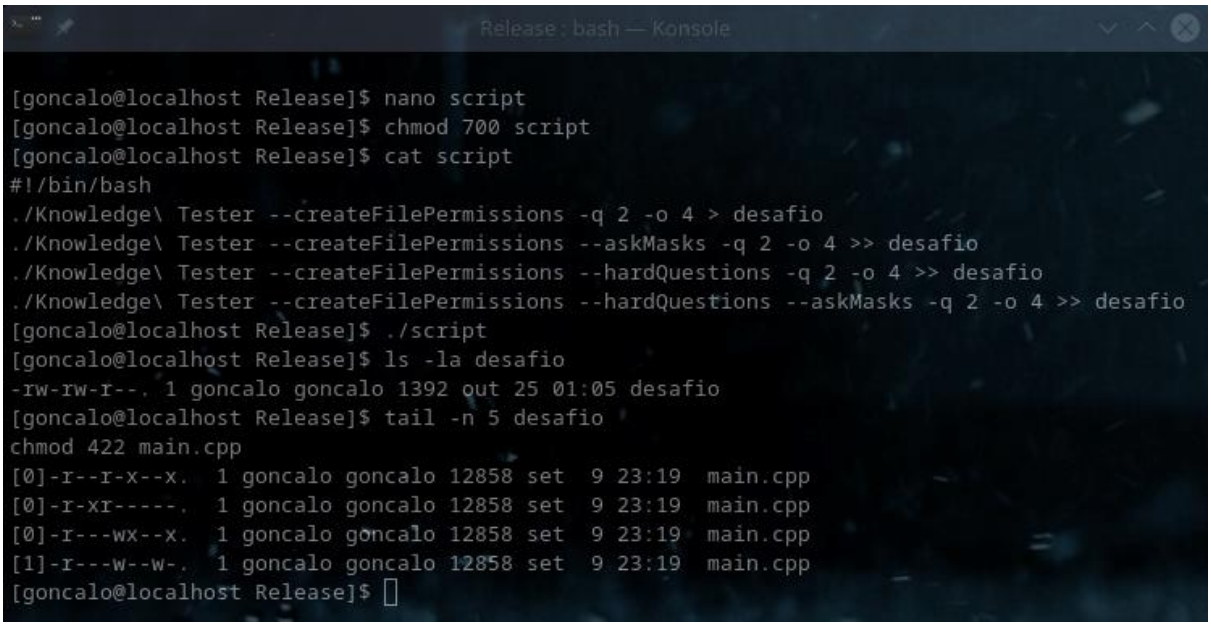
Teremos de usar quatro comandos distintos, para adicionar as perguntas ao mesmo ficheiro, e podemos até criar um script que faça isto por nós, mas os comandos são:

- ./Knowledge\ Tester --createFilePermissions -q 2 -o 4 > desafio
- ./Knowledge\ Tester --createFilePermissions --askMasks -q 2 -o 4 >> desafio
- ./Knowledge\ Tester --createFilePermissions --hardQuestions -q 2 -o 4 >> desafio
- ./Knowledge\ Tester --createFilePermissions --hardQuestions --askMasks -q 2 -o 4 >> desafio

Com a primeira linha, criamos o ficheiro "desafio", e se já existir, é limpo primeiro e guardadas lá as perguntas depois, porque temos um único sinal de ">" após a linha de comandos.

Com as restantes linhas, já temos não um mas dois sinais de maior (">>"), o que faz com que as perguntas sejam adicionadas ao fim do ficheiro "desafio" já existente ao invés de o sobrescrever.

Com estas quatro linhas criamos um *script* que trata da tarefa sempre que quisermos:



Criamos o script com o que podemos ver acima (visualizado o seu conteúdo com o comando "*cat script*" acima, e após lhe darmos privilégios de execução com um "*chmod 700 script*", é executado com um "*./script*", e o ficheiro "desafio" é criado com as perguntas e respostas que podem ver acima.

E assim podem juntar vários exercícios destes num só ficheiro.

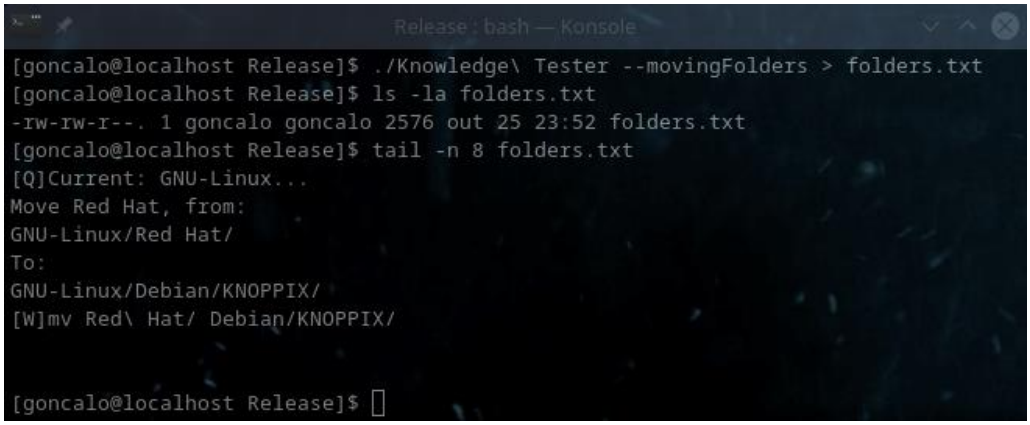
Exercícios de Mover Pastas

Este *software* tem incluído uma funcionalidade de geração de exercícios de mover pastas, para o utilizador se adaptar a guiar-se através de caminhos relativos ao copiar e mover pastas e ficheiros.

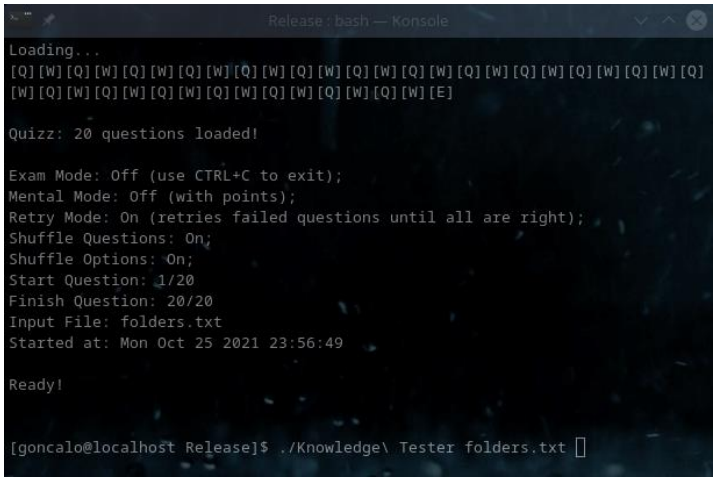
O objectivo é que o utilizador ganhe em poucos minutos a prática que levaria largas horas a ganhar, ou até dias, numa máquina real (pois numa máquina real não estaria a mover dezenas de pastas como está neste exercício), e assim acelera a aprendizagem como acontece com o exercício das *file permissions*.

Como criar exercício de mover pastas

Esse exercício é gerado através da *tag* "*--movingFolders*", e terá obviamente de ter adicionado ao fim da linha de comandos um "*> fich.txt*", onde "*fich.txt*" é o nome do ficheiro que vai conter o exame:



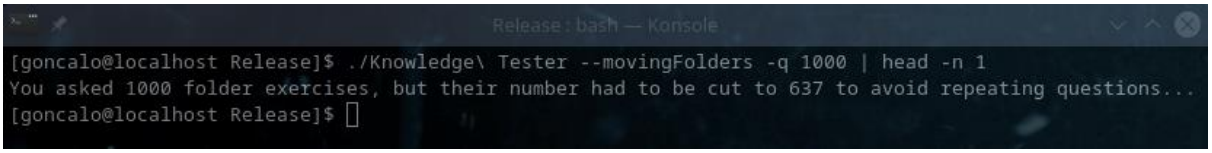
A partir desse momento, corremos o exercício como qualquer outro, com um simples "*./Knowledge\ Tester fich.txt*", ou neste caso "*folders.txt*":



Escusado será dizer que cada vez que a *tag* "*--movingFolders*" é chamada, os exercícios gerados serão completamente diferentes dos anteriores, e "aleatórios" (dentro do possível).

Opções disponíveis

Neste exercício existe a opção de alterar o número de perguntas a serem geradas, através da *tag* "*-n*", mas que está limitado ao número de 637 porque com um número superior a esse, perguntas repetir-se-iam:

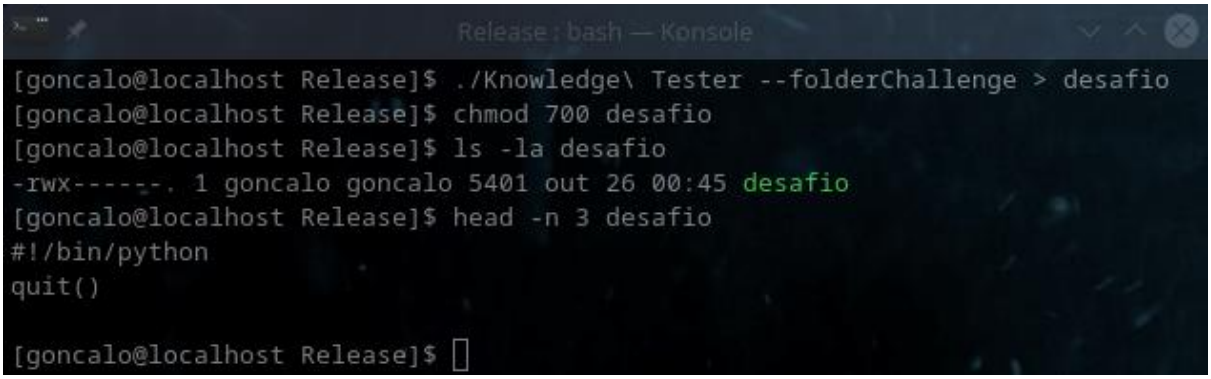


Esta é a única opção de momento disponível neste exercício...

Desafio de Pastas "*Folder Challenge*"

Este é um exercício mais complexo e trabalhoso, e que envolve não só prática a mudar nomes a pastas e movê-las, mas também algum ligeiro "*troubleshooting*", pois obriga o utilizador a resolver um pequeno problema, além de andar a verificar que pastas estão no local errado ou com nome errado, sendo um trabalho talvez mais aborrecido mas eficaz.

Este exercício invoca-se com um "*--folderChallenge*", e temos, tal como nos outros, de reencaminhar o *output* do *software* para um ficheiro de texto com o uso do sinal "*>*" no fim da linha, neste caso um *script* de nome "desafio":



O processo inicial é o descrito acima, o envio do *output* do comando para o *script* de nome "desafio", dar permissões de execução ao mesmo (com o "*chmod 700*" acima), e podemos ver com o comando "*head -n 3*" do exemplo acima, que o *script* é criado.

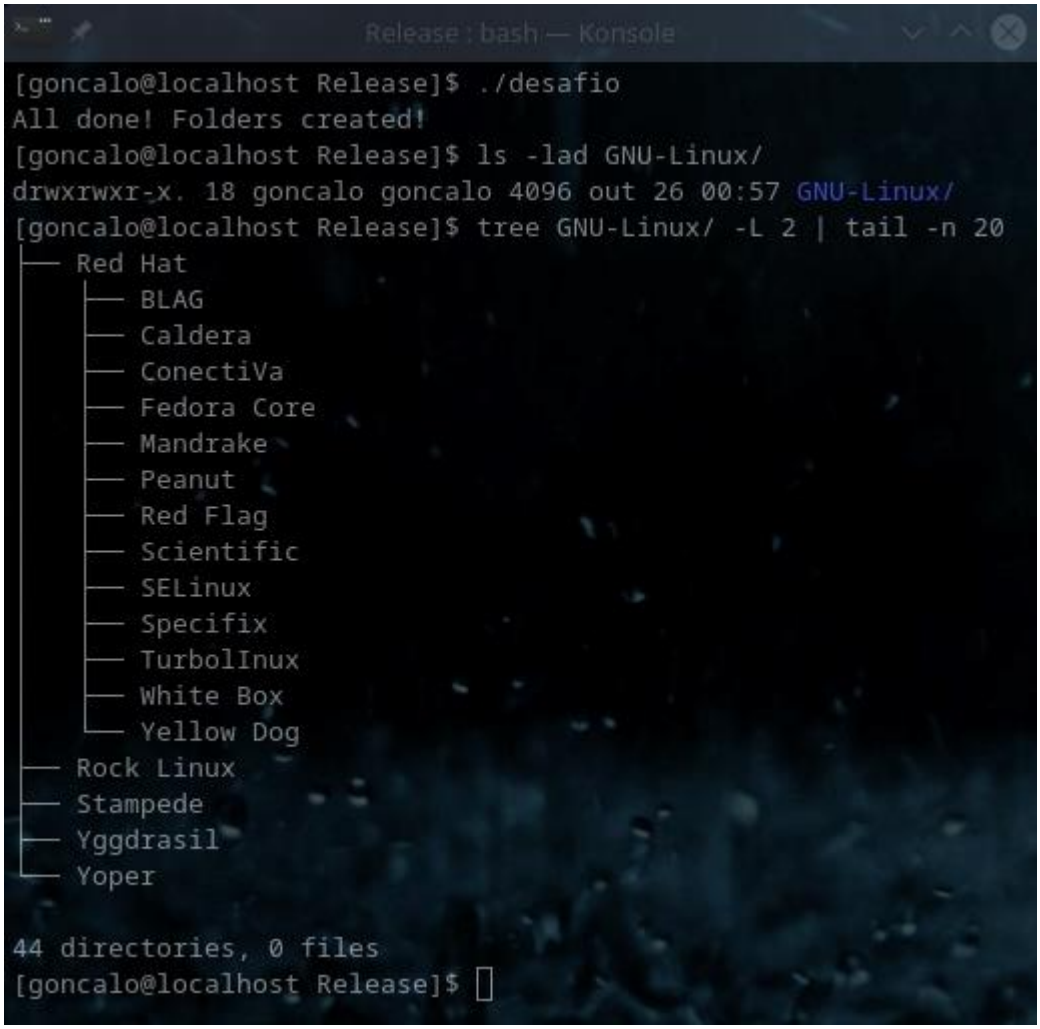
O erro de *scripts* a ser corrigido

Mas à partida tem logo dois erros que devem ser corrigidos:

- A primeira linha deveria ter "*#!/bin/bash*" e não "*#!/bin/python*", pois o *script* é de *bash* e não de *python*. Esta linha foi incluída num exercício inicial, e acabou por tradicionalmente ficar cá, para fazer os formandos compreender para que servem.
- A linha que contém o "*quit()*" deve ser apagada, pois é um comando de *python* indesejado ao *script bash* que vamos executar.

A árvore de pastas criadas

Depois de corrigidos os erros, o *script* é executado com um simples "*./desafio*", e cria-nos de imediato uma pasta cheia de perto de 90 directorias com nomes de distribuições de *Linux* famosas e até as suas hierarquias, que serão as usadas para este exercício:



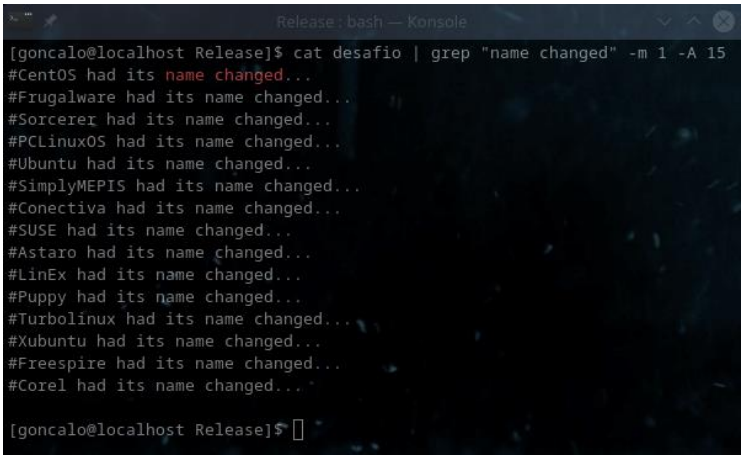
Acima podemos ver apenas até dois níveis de profundidade (com um "*head -L 2*") e as últimas 20 linhas (com um "*tail -n 20*") da árvore das perto de 90 pastas de *distros Linux*.

Podemos ver também pelas pastas acima que o *script* foi criado de forma a não só criar as pastas, como mudar o nome a várias delas (como podem ver pela imagem acima onde várias têm os seus nomes alterados) e até mover pastas de sítio.

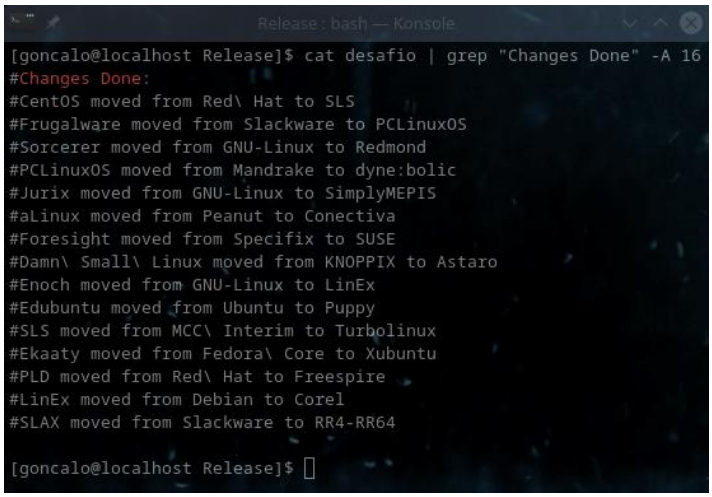
O objectivo de quem resolve o exercício é o de mudar os nomes das pastas para os nomes originais, e mover as pastas deslocadas para as suas posições originais, e isto obriga a algum treino forçado nestas tarefas.

As ajudas dadas aos formandos

O próprio *script*, como forma de ajuda aos alunos, tem a lista de pastas que mudaram de nome, para evitar que percam tempo a tentar descobrir quais foram:



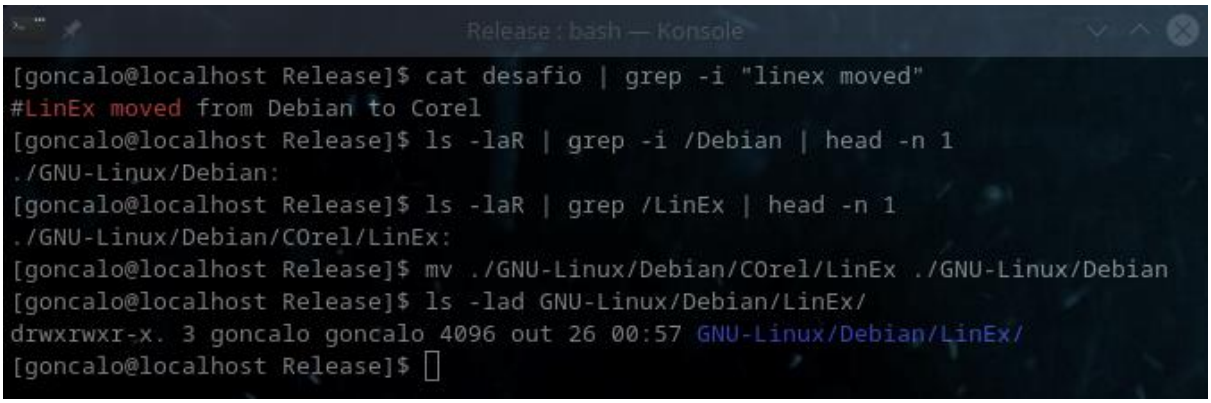
E da mesma forma, estão no ficheiro "desafio" também incluídas as mudanças de localização das pastas que foram deslocadas, para que os utilizadores saibam quais devem corrigir, onde só terão de mover da posição actual para a antiga localização:



O utilizador pode poupar tempo, com uma pesquisa rápida, com um "*grep -i*", cujo "*-i*" torna a busca "*case insensitive*" e com isso, resistente às alterações de minúsculas para maiúsculas (e vice-versa), e com uma *regular expression* de "*^GNU.*/Linex:$*", que significará algo como "linhas começadas em *GNU*, com qualquer coisa no meio, e terminadas em *Linex*:" (mas *case insensitive*), e surgirá a localização da mesma, e com um simples *move* (comando "*mv*"), é de imediato corrigida a situação:

O mover pastas de sítio, deve ser efectuado depois, e a começar de baixo para cima, para evitar problemas, pois se for noutra ordem o exercício provavelmente irá falhar.

Nessa correcção de localização de pastas, o raciocínio é simples: além de começar do último para o primeiro, só temos de localizar a actual posição da pasta a mover, que é a "*LinEx*" neste caso, e a localização da antiga pasta de origem, que neste caso era a "*Debian*", e depois mover da actual para a antiga, é tão simples quanto isso:



E repetir os passos com todas as outras.

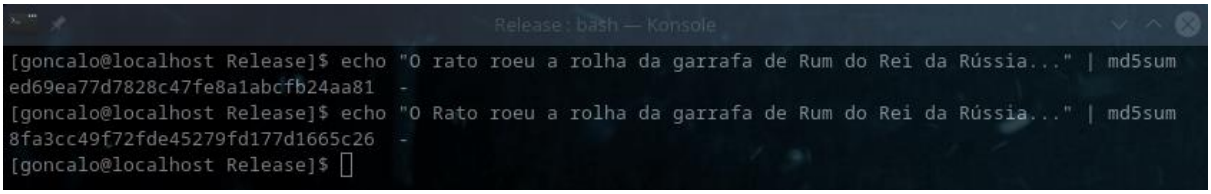
As hashes como sistema de verificação de resultados

Agora, como verificar se tudo ficou correcto?

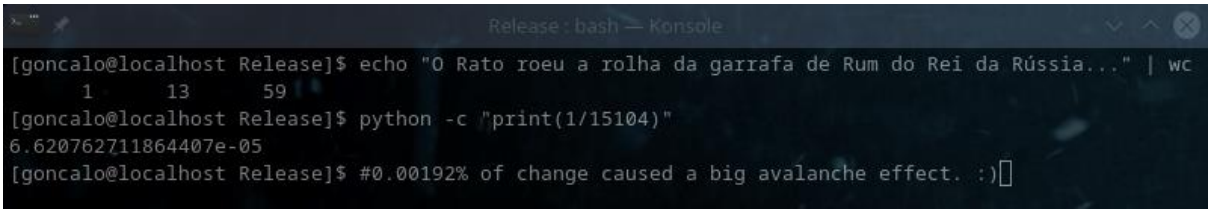
O sistema usado neste caso foi muito simples, o uso de um sistema de *hashing* criptográfico, já quebrado há muito mas mais do que suficiente para o exercício em questão, o *MD5*.

Sendo uma *hash* criptográfica, tem o chamado "efeito de avalanche", que faz com que, por muito ínfima que seja a alteração no texto que dá origem à *hash*, as repercussões na *hash* em si são sempre enormes, daí o nome "efeito avalanche".

Vejamos abaixo o "efeito de avalanche" em acção:

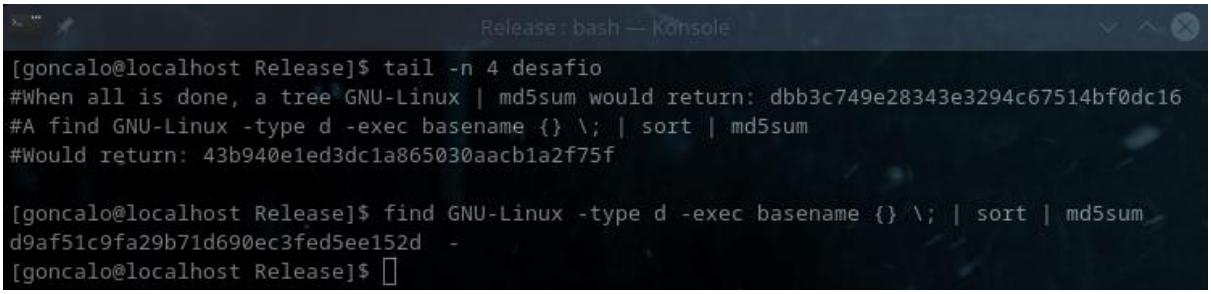


Basicamente, nestas 59 letras, uma alteração de um *bit* apenas, causaria uma alteração total da *hash MD5*, e ainda mais se for não um *bit* mas de 32, que corresponde à alteração da letra "r" para "R" no código *ASCII*:



Isto significa que uma simples letra alterada no meio dos nomes destas perto de 90 pastas que correspondem a umas 90 *distros* famosas de *Linux*, alteraria por completo a *hash*, quanto mais se estiverem pastas fora do local, etc.

Para confirmarmos, corremos o comando que é indicado no fim do ficheiro "desafio" (nome dado neste caso):



Com o comando "*find -type d -exec basename {} \;*" procuramos directorias, e temos como resultado apenas o nome das mesmas sem nada mais (graças ao comando "*basename {} \;*"), que depois são ordenados com o comando "*sort*", e por fim gerada a *hash* correspondente.

Não sendo uma *hash* perceptual, não dá para saber a quantidade de erros que existirão pela *hash*, pois à mínima coisa, ela muda radicalmente, mas o aluno/formando terá de investigar qual o problema, até ter o resultado correcto.

E assim termina a ajuda sobre o terceito exercício embutido neste *software* de aprendizagem.

O sistema de *Tags*

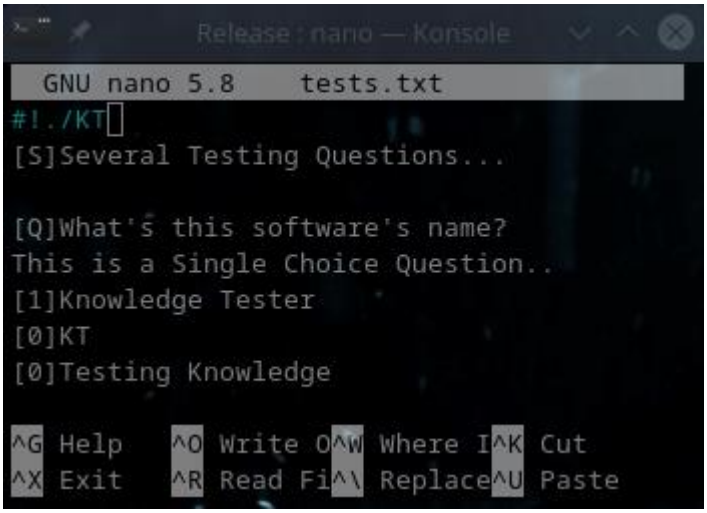
Este *software* é muito baseado num sistema de *tags*, existindo tags como [Q], [S], [W], [1], [0], [I], [L], e mais tarde serão adicionadas outras, mas de forma simples, a proporcionar que qualquer pessoa possa criar exames em puro texto, num simples bloco de notas.

Mais tarde serão adicionadas outras à medida que o *software* vá sendo desenvolvido.

Como transformar um exame num executável

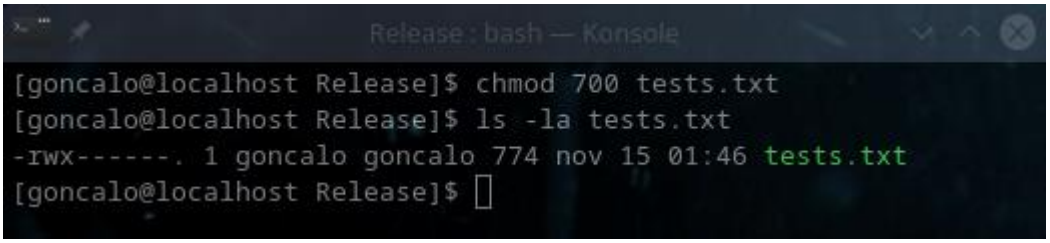
Tal como sucede com *scripts* em *bash* ou *python* ou outros, podemos mandar executar um exame como se fosse um ficheiro executável, se não for encriptado.

Só temos de no começo colocar no lugar usado para definir o interpretador do *script* o caminho para o executável do *Knowledge Tester*, evitando espaços nos nomes:

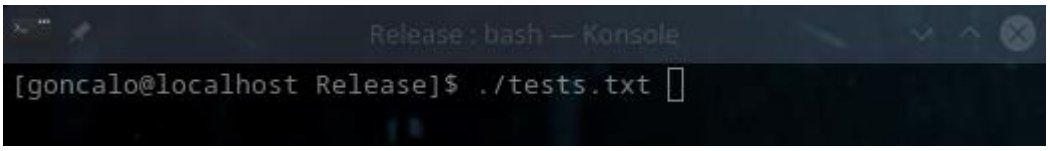


No exemplo acima, definimos o interpretador do exame como sendo o executável de nome "*KT*".

Depois temos de dar permissões de executável ao ficheiro do exame em questão, com um simples "*chmod 700*":



Depois já poderemos executar o comando, desde que haja um *software Knowledge Tester* com o nome "*KT*" na pasta em questão, claro está, e ele correrá como se fosse um ficheiro executável:



É apenas uma ideia que poderá ser útil para facilitar algumas operações, nada mais.

Ideias Futuras

Provavelmente este *software* irá ficar pendente durante uns bons meses, à medida que regressarei a projectos parados há já algum tempo, como o meu *Game Engine* em *C/C++*, mas mais tarde hei-de voltar a ele.

Mas antes de ser colocado em *stand-by*, tenciono ainda em 2021:

- Criar um sistema de opções forçadas, para que possamos gerar um exame que corra obrigatoriamente com certas opções activadas e que seja impossível desligar com parâmetros na linha de comandos;
- Criar um sistema (talvez) de respostas múltiplas por cada linha, ou seja, numa resposta multi-linha, permitir que certas linhas tenham várias respostas possíveis, talvez ainda este ano, ou talvez fique para o próximo;
- Poder definir a pontuação de cada pergunta, ao invés de ser como agora em que cada pergunta valerá o valor de 100% a dividir pelo número de perguntas total;

  De resto, para 2021, parece-me mais do que completo de momento.

No Futuro, mais tarde, pretendo:

- Criar talvez uma *GUI*, para fazer exames em modo gráfico, a imitar certificações;

- Talvez possibilitar a exibição de imagens em exames, mesmo em modo de texto (alterando directamente os *buffers* de memória usados pelo *Linux* para vídeo) em modo de terminal;
- Acrescentar outro tipo de perguntas e respostas, como ordenações, etc;
- E outras coisas que não me ocorrem de momento.

Contactos e Informações

Em caso de críticas construtivas, ou sugestões, ou dúvidas, contactem-me quem me conhece pelos meios tradicionais, e quem não me conhece, pelo *LinkedIn* ou *email*, disponíveis no meu *website* *http://www.goncalo.pt/*.