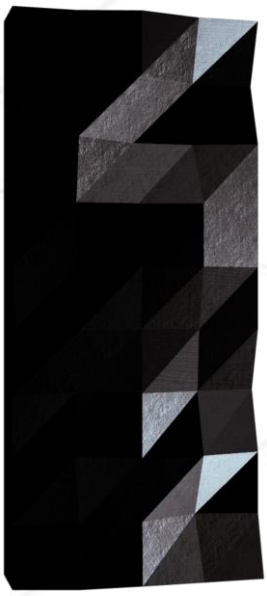

Minishell

Tão bela como uma concha

Resumo:
Este projecto trata da criação de uma simples concha.
Sim, a sua própria pequena festa.
Aprenderá muito sobre processos e descritores de ficheiros.

Versão: 6

Conteúdos

Capítulo I

Introdução

A existência de conchas está ligada à própria existência das TI.

Na altura, todos os programadores concordaram que *a comunicação com um computador utilizando interruptores 1/0 alinhados era seriamente irritante.*

Era apenas lógico que tivessem a ideia de criar um software para comunicar com um computador utilizando linhas interactivas de comandos numa língua um pouco próxima da língua humana.

Graças ao Minishell, poderá viajar através do tempo e voltar a problemas que as pessoas enfrentaram quando *o Windows* não existia.

Capítulo II

Instruções comuns

- O seu projecto deve ser escrito em C.
- O seu projecto deve ser escrito em conformidade com a Norma. Se tiver ficheiros/funções de bónus, estes são incluídos na verificação da norma e receberá um 0 se houver um erro na norma.
- As suas funções não devem desistir inesperadamente (falha de segmentação, erro de autocarro, duplo livre, etc.) para além de comportamentos indefinidos. Se isto acontecer, o seu projecto será considerado não funcional e receberá um 0 durante a avaliação.
- Todo o espaço de memória atribuído ao monte deve ser devidamente libertado quando necessário. Não serão toleradas fugas de informação.
- Se o assunto o exigir, deve submeter um Makefile que compilará os seus ficheiros de origem à saída requerida com as bandeiras -Wall, -Wextra e -Werror, usar cc, e o seu Makefile não deve voltar a ligar-se.
- O seu Makefile deve pelo menos conter as regras $(NOME), tudo, limpo, limpo, limpo e
  re.
- Para entregar bónus ao seu projecto, deve incluir um bónus de regra ao seu Makefile, que acrescentará todos os vários cabeçalhos, librairies ou funções que são proibidas na parte principal do projecto. Os bónus devem estar num ficheiro diferente _bonus.{c/h} se o assunto não especificar mais nada. A avaliação obrigatória e da parte de bónus é feita separadamente.
- Se o seu projecto lhe permitir utilizar a sua libft, deverá copiar as suas fontes e o seu Makefile associado numa pasta libft com o seu Makefile associado. O Makefile do seu projecto deve compilar a biblioteca utilizando o seu Makefile, depois compilar o projecto.
- Encorajamo-lo a criar programas de teste para o seu projecto, embora este trabalho **não tenha de ser submetido e não tenha de ser classificado.** Dar-lhe-á a oportunidade de testar facilmente o seu trabalho e o trabalho dos seus pares. Encontrará esses testes especialmente úteis durante a sua defesa. De facto, durante a defesa, é livre de utilizar os seus testes e/ou os testes dos seus pares que está a avaliar.
- Submeta o seu trabalho ao seu repositório de git atribuído. Apenas o trabalho no repositório de git - tory será classificado. Se o Deepthought for designado para

classificar o teu trabalho, será feito

após as suas avaliações pelos pares. Se acontecer um erro em qualquer secção do seu trabalho durante a avaliação do Deepthought, a avaliação irá parar.

Capítulo III

Parte Obrigatória

Nome do programa	mini-concha
Entrega de ficheiros	Makefile, *.h, *.c
Makefile	NOME, tudo, limpo, limpo, fclean, re
Argumentos	
Funcionalidades externas.	readline, rl_clear_history, rl_on_new_line, rl_replace_line, rl_redisplay, add_history, printf, malloc, free, write, access, open, read, close, fork, wait, waitpid, wait3, wait4, signal, sigaction, sigemptyset, sigaddset, kill, exit, getcwd, chdir, stat, lstat, fstat, unlink, execve, dup, dup2, pipe, opendir, readdir, closedir, strerror, perror, isatty, ttyname, ttyslot, ioctl, getenv, tcsetattr, tcgetattr, tgetgetlag, tgetnum, tgetstr, tgoto, tputs
Libft autorizado	Sim
Descrição	Escrever uma concha

A sua concha deve:

- Mostrar um **prompt** quando se espera por um novo comando.
- Ter uma **história de** trabalho.
- Pesquisar e lançar o executável correcto (com base na variável PATH ou utilizando um caminho relativo ou absoluto).
- Não utilizar mais do que **uma variável global**. Pense sobre isso. Terá de explicar a sua finalidade.
- Não interpretar citações não fechadas ou caracteres especiais que não são exigidos pelo sujeito, tais como \i1 (barra invertida) ou `;` (ponto e vírgula).
- Manusear " (citação única) que deve impedir a concha de interpretar os meta-caracteres na sequência citada.
- Punho " (aspa dupla) que deve impedir a concha de interpretar os meta- caracteres na sequência citada, excepto $ (sinal de dólar).

- Implementar **redireccionamentos**:
  - < deve redireccionar a entrada.
  - > deve redireccionar a produção.
  - << deve ser dado um delimitador, depois ler a entrada até ser vista uma linha contendo o delimitador. No entanto, não tem de actualizar a história!
  - >> deve redireccionar a saída em modo anexo.
- Implementar **tubos** (| carácter). A saída de cada comando na tubagem é ligada à entrada do próximo comando através de uma tubagem.
- Manusear **variáveis de ambiente** ($ seguidos de uma sequência de caracteres) que devem expandir-se para os seus valores.
- Manusear $? que deverá expandir-se para o estado de saída do gasoduto em primeiro plano executado mais recentemente.
- Manusear ctrl-C, ctrl-D e ctrl-\ que devem comportar-se como em bash.
- Em modo interactivo:
  - ctrl-C exibe um novo prompt numa nova linha.
  - ctrl-D sai da casca.
  - ctrl-\i1 não faz nada.
- A sua concha deve implementar as seguintes **construções**:
  - echo com opção -n
  - cd com apenas um caminho relativo ou absoluto
  - pwd sem opções
  - exportação sem opções
  - sem opções
  - inveja sem opções ou argumentos
  - sair sem opções

A função readline() pode causar fugas de memória. Não é necessário corrigi-los. Mas isso **não significa que o seu próprio código, sim o código que escreveu, pode causar fugas de memória**.

Deve limitar-se à descrição do assunto. Tudo o que não for pedido não é necessário.
Se tiver alguma dúvida sobre um requisito, tome a bash como referência.

Capítulo IV

Parte Bónus

O seu programa tem de ser implementado:

- && e ||| com parênteses para prioridades.
- Os wildcards * devem funcionar para o directório de trabalho actual.

A parte de bónus só será avaliada se a parte obrigatória for PERFEITA. Perfeito significa que a parte obrigatória foi feita integralmente e funciona sem avarias. Se não tiver passado TODOS os requisitos obrigatórios, a sua parte de bónus não será de todo avaliada.

Capítulo V

Submissão e avaliação pelos pares

Entregue a sua tarefa no seu repositório Git como de costume. Apenas o trabalho dentro do teu repositório será avaliado durante a defesa. Não hesites em verificar duas vezes os nomes dos teus ficheiros para garantir que estão correctos.