Miniconversa

Resumo:
O objetivo deste projeto é codificar um pequeno programa de troca de dados usando sinais UNIX.

Versão: 3

Conteúdo

Capítulo I

Prefácio

O cis-3-Hexen-1-ol, também conhecido como (Z)-3-hexen-1-ol e álcool foliar, é um líquido oleoso incolor com um intenso odor verde-gramado de grama e folhas verdes recém-cortadas.

É produzido em pequenas quantidades pela maioria das plantas e atua como atrativo para muitos insetos predadores. cis-3-Hexen-1-ol é um composto aromático muito importante usado em sabores de frutas e vegetais e em perfumes.

A produção anual é de cerca de 30 toneladas.

Capítulo II

Instruções Comuns

- Seu projeto deve ser escrito em C.
- Seu projeto deverá ser escrito de acordo com a Norma. Se você tiver arquivos/funções bônus, eles serão incluídos na verificação de norma e você receberá 0 se houver um erro de norma.
- Suas funções não devem encerrar inesperadamente (falha de segmentação, erro de barramento, liberação dupla, etc.) além de comportamentos indefinidos. Caso isso aconteça, seu projeto será considerado não funcional e receberá nota 0 na avaliação.
- Todo o espaço de memória alocado no heap deve ser liberado adequadamente quando necessário. Sem vazamentos será tolerado.
- Se o assunto exigir, você deve enviar um Makefile que irá compilar seus arquivos fonte para a saída necessária com as flags -Wall, -Wextra e -Werror, use cc, e seu Makefile não deve relinkar.
- Seu Makefile deve conter pelo menos as regras $(NAME), all, clean, fclean e ré.
- Para entregar bônus ao seu projeto, você deve incluir uma regra bônus em seu Makefile, que adicionará todos os diversos cabeçalhos, bibliotecas ou funções que são proibidas na parte principal do projeto. Os bônus devem estar em um arquivo diferente _bonus.{c/h} se o assunto não especificar mais nada. A avaliação da parte obrigatória e da parte bônus é feita separadamente.
- Se o seu projeto permitir que você use sua libft, você deve copiar seus fontes e seu Makefile associado em uma pasta libft com seu Makefile associado. O Makefile do seu projeto deve compilar a biblioteca usando seu Makefile e depois compilar o projeto.
- Nós encorajamos você a criar programas de teste para o seu projeto, mesmo que este trabalho **não precise ser enviado e não receba notas.** Isso lhe dará a chance de testar facilmente seu trabalho e o de seus colegas. Você achará esses testes especialmente úteis durante sua defesa. Na verdade, durante a defesa, você é livre para usar seus testes e/ou os testes do colega que está avaliando.
- Envie seu trabalho para o repositório git designado. Somente o trabalho no repositório git será avaliado. Se Deepthought for designado para avaliar seu trabalho, isso será feito

após suas avaliações por pares. Se ocorrer um erro em qualquer seção do seu trabalho durante a avaliação do Deepthought, a avaliação será interrompida.

Capítulo III

Instruções do projeto

- Nomeie seus arquivos executáveis como cliente e servidor.
- Você terá que entregar um Makefile que irá compilar seus arquivos fonte. Não deve vincular novamente.
- Você definitivamente pode usar seu libft.
- Você precisa lidar com os erros minuciosamente. De forma alguma seu programa deve sair inexplicavelmente corretamente (falha de segmentação, erro de barramento, liberação dupla e assim por diante).
- Seu programa não deve ter **vazamentos de memória.**
- Você pode ter **uma variável global por programa** (uma para o cliente e outra para o servidor), mas você terá que justificar seu uso.
- Para completar a parte obrigatória, você pode usar o seguinte funções:
  - ÿ escrever
  - ÿ ft_printf e qualquer equivalente codificado por VOCÊ
  - ÿ sinal
  - ÿ sigemptyset
  - ÿ sigaddset ÿ
  - sigaction
  - ÿ matar
  - ÿ getpid
  - ÿ maloc
  - ÿ grátis
  - ÿ pausa
  - ÿ dormir
  - ÿ dormir
  - ÿ sair

Capítulo IV

Parte Obrigatória

Você deve criar um programa de comunicação na forma de **cliente** e **servidor.**

- O servidor deve ser iniciado primeiro. Após o seu lançamento, deverá imprimir seu PID.
- O cliente usa dois parâmetros:
  - ÿ O PID do servidor.
  - ÿ A string a ser enviada.
- O cliente deve enviar a string passada como parâmetro ao servidor.
  Assim que a string for recebida, o servidor deverá imprimi-la.
- O servidor precisa exibir a string rapidamente. Rapidamente significa que se você pensar demora muito, então provavelmente é muito longo.

1 segundo para exibir 100 caracteres é **demais!**

- Seu servidor deverá ser capaz de receber strings de vários clientes consecutivos sem precisar reinicializar.
- A comunicação entre seu cliente e seu servidor deve ser feita **apenas** usando Sinais UNIX.
- Você só pode usar estes dois sinais: SIGUSR1 e SIGUSR2.

O sistema Linux NÃO enfileira sinais quando você já possui sinais pendentes deste tipo! Tempo de bônus?

Capítulo V

Parte bônus

Lista de bônus:

- O servidor reconhece cada mensagem recebida enviando de volta um sinal para o cliente.
- Suporte a caracteres Unicode!

A parte bônus só será avaliada se a parte obrigatória for PERFEITA. Perfeito significa que a parte obrigatória foi feita integralmente e funciona sem mau funcionamento. Se você não passou em TODOS os requisitos obrigatórios, sua parte do bônus não será avaliada de forma alguma.

Capítulo VI

Envio e avaliação por pares

Entregue sua tarefa em seu repositório Git normalmente. Somente o trabalho dentro do seu repositório será avaliado durante a defesa. Não hesite em verificar os nomes dos seus arquivos para garantir que estão corretos.

arquivo.bfe:VAAe8ElCrUAbXivz0ueiIpv/u/ia9PL50+HI+8/bgPKLESHlp
tPLpu0PW9zWV/LwDVaOqCRCGu6Gopk1X0i6Kn7t