Pipex

Resumo:
Este projeto permitirá que você descubra em detalhes um mecanismo UNIX que você já conhece usando-o em seu programa.

Versão 2

Conteúdo

Capítulo I

Prefácio

Cristina: "Vá dançar salsa em algum lugar :)"

Capítulo II

Instruções Comuns

- Seu projeto deve ser escrito em C.
- Seu projeto deve ser redigido de acordo com a Norma. Se você tiver arquivos/funções de bônus, eles serão incluídos na verificação da norma e você receberá um 0 se houver um erro de norma dentro.
- Suas funções não devem ser encerradas inesperadamente (falha de segmentação, erro de barramento, double free, etc.) além de comportamentos indefinidos. Se isso acontecer, 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 de origem para a saída necessária com os sinalizadores -Wall, -Wextra e -Werror, use cc e seu Makefile não deve revincular.
- Seu Makefile deve conter pelo menos as regras $(NAME), all, clean, fclean e ré.
- Para ativar bônus em seu projeto, você deve incluir uma regra de bônus em seu Makefile, que adicionará todos os vários 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 bônus é feita separadamente.
- Se seu projeto permite que você use sua libft, você deve copiar suas 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, em seguida, compilar o projeto.
- Incentivamos você a criar programas de teste para o seu projeto, mesmo que este trabalho **não precise ser enviado e não seja avaliado.** Isso lhe dará a chance de testar facilmente seu trabalho e o de seus colegas. Você achará esses testes especialmente úteis durante sua defesa. De fato, durante a defesa, você é livre para usar seus testes e/ou os testes do par que está avaliando.
- Envie seu trabalho para o repositório git atribuído. Somente o trabalho no repositório git será avaliado. Se o Deepthought for designado para avaliar seu trabalho, isso será feito

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

Capítulo III

parte obrigatória

Nome do programa	pipex
Entregar arquivos	Makefile, *.h, *.c NAME, all,
Argumentos	clean, fclean, re file1 cmd1 cmd2 file2
Makefile	
Funções externas.	• abrir, fechar, ler, escrever, malloc, free, perror, strerror, access, dup, dup2, execve, exit, fork, pipe, unlink, wait, waitpid • ft_printf e qualquer equivalente VOCÊ codificou
Liberação autorizada	Sim
Descrição	Este projeto é sobre o manuseio de tubos.

Seu programa será executado da seguinte forma:

./pipex arquivo1 cmd1 cmd2 arquivo2

Deve ter 4 argumentos:

• arquivo1 e arquivo2 são **nomes de arquivo.**

• cmd1 e cmd2 são **comandos shell** com seus parâmetros.

Ele deve se comportar exatamente da mesma forma que o comando shell abaixo:

```
$> < arquivo1 cmd1 | cmd2 > arquivo2
```

III.1 Exemplos

```
$> ./pipex arquivo de saída "ls -l" "wc -l"
```

Deve se comportar como: < infile ls -l | wc -l > arquivo de saída

```
$> ./pipex infile "grep a1" "wc -w" outfile
```

Deve se comportar como: < infile grep a1 | wc -w > arquivo de saída

III.2 Requisitos

Seu projeto deve obedecer as seguintes regras:

- Você tem que entregar um Makefile que irá compilar seus arquivos fonte. não deve religar.
- Você tem que lidar com os erros completamente. De maneira alguma seu programa deve sair do unex especificado (falha de segmentação, erro de barramento, double free, e assim por diante).
- Seu programa não deve ter **vazamentos de memória.**
- Em caso de dúvida, trate os erros como o comando shell: < arquivo1 cmd1 | cmd2 > arquivo2

Capítulo IV

Parte bônus

Você receberá pontos extras se:

- Manuseie vários tubos.

  Este:

  ```
  $> ./pipex arquivo1 cmd1 cmd2 cmd3 ... cmdn arquivo2
  ```

  Deve se comportar como:

  ```
  < arquivo1 cmd1 | cmd2 | cmd3... | cmdn > arquivo2
  ```

- Suporte « e » quando o primeiro parâmetro é "here_doc".

  Este:

  ```
  $> ./pipex here_doc LIMITER cmd cmd1 arquivo
  ```

  Deve se comportar como:

  ```
  cmd << LIMITADOR | cmd1 >> arquivo
  ```

A parte bônus só será avaliada se a parte obrigatória for PERFEITA. Perfeito significa que a peça obrigatória foi executada integralmente e funciona sem avarias. Se você não passou em TODOS os requisitos obrigatórios, sua parte de bônus não será avaliada.

Capítulo V

Submissão e avaliação por pares

Entregue sua atribuição em seu repositório Git como de costume. Apenas o trabalho dentro do seu repositório será avaliado durante a defesa. Não hesite em verificar novamente os nomes de seus arquivos para garantir que estejam corretos.