Sistemas Distribuídos ÿ (https://www.cs.columbia.edu/~du/ds)

Lição de casa (https://www.cs.columbia.edu/~du/ds/homework.html)

Dicas (https://www.cs.columbia.edu/~du/ds/tips.html)

Políticas (https://www.cs.columbia.edu/~du/ds/policies.html)

Agenda (https://www.cs.columbia.edu/~du/ds/schedule.html)

Discussão (https://piazza.com/columbia/fall2013/w49952/home)

Trabalho de casa1 C++ Aquecimento

Prazo: 2013/09/11 23:59:59

1344 dias se passaram

NOTA: Este primeiro dever de casa é classificado apenas como Aprovado/Reprovado

ÿ Problemas escritos

ÿ Problemas de programação

O que você precisa enviar no final?

Ao final da lição de casa, você precisa enviar os seguintes arquivos

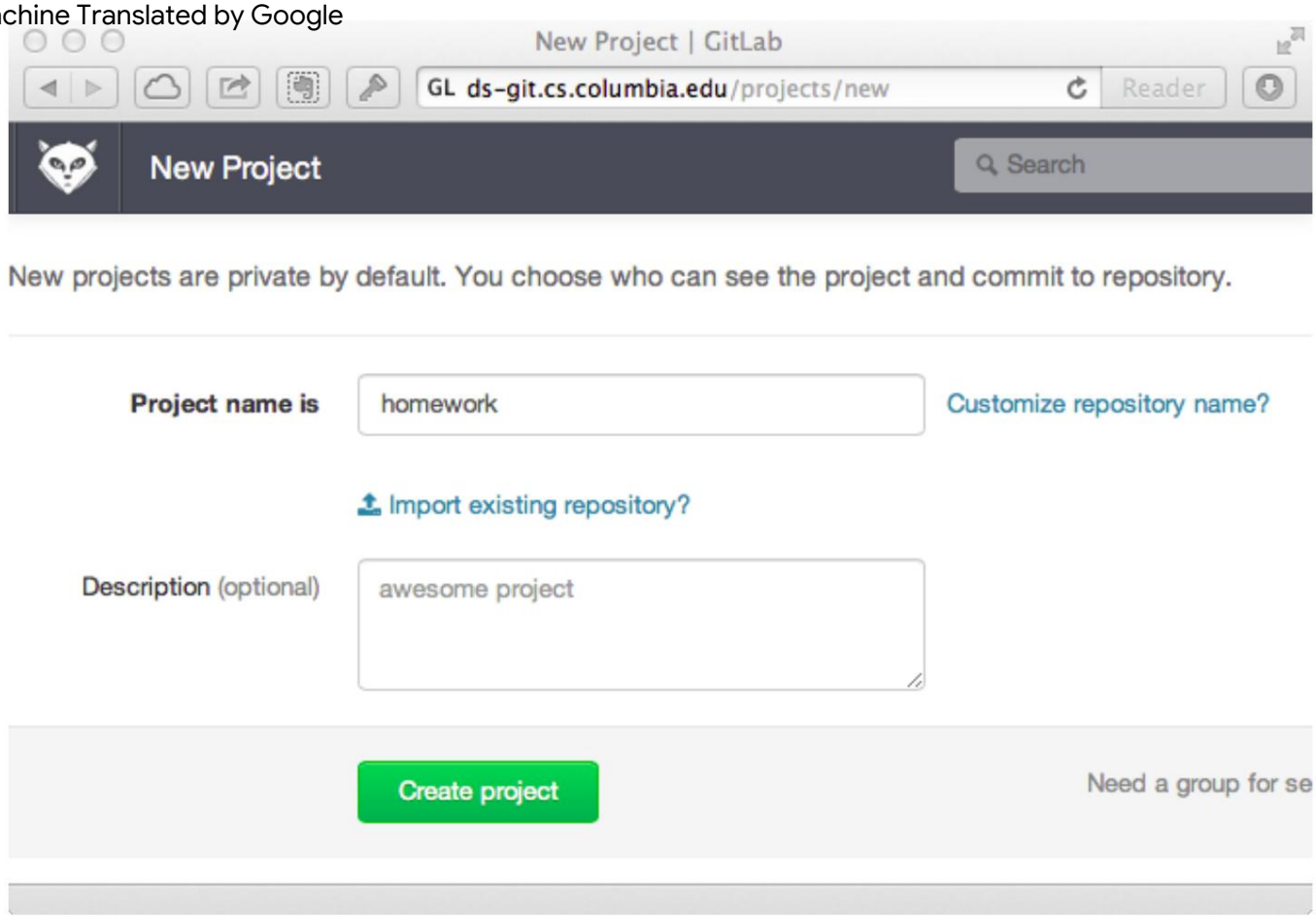
1. Um arquivo de texto chamado `ESCRITO` que é para os problemas escritos
2. `Arquivo` README que explica o que você fez para a lição de casa de programação, incluindo os casos de teste
3. Todo o seu código fonte para problemas de programação 4.

Um `makefile` que gera tcp_client e `tcp_server` do seu código `fonte`

Como criar repositório e enviar via Git?

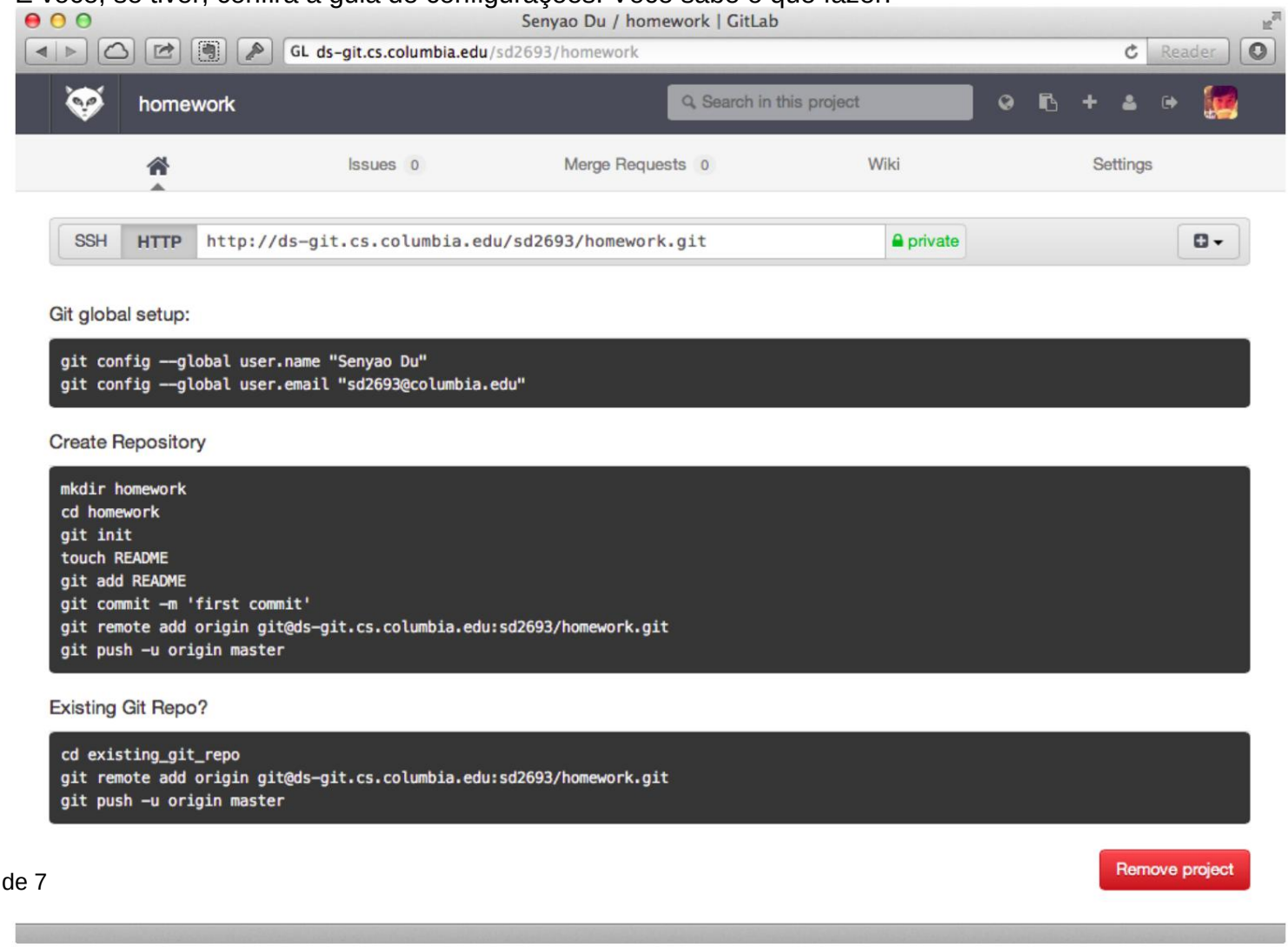
Esta é a primeira tarefa, e eu vou trabalhar com você em toda a configuração para as tarefas que estão por vir

1. Faça login em `ds-git.cs.columbia.edu` usando sua conta CS e visite a página New Project (http://ds-git.cs.columbia.edu/projects/new) para criar um projeto chamado `dever de casa`



2. Siga as instruções na tela para um novo repositório Git. Use o endereço HTTP se você não tiver chaves SSH.

E você, se tiver, confira a guia de configurações. Você sabe o que fazer.



Você deve ter uma pasta chamada homework agora. 3.

```
% CD dever de casa
% git checkout -b homework1
```

após esses dois comandos, você está pronto para ir, código para este projeto.

Depois de ter feito com o projeto 4.

```
% git commit -am "Trabalho de casa concluído 1" % git push
```

Lembre-se de enviar suas alterações de código. Envie os arquivos para o servidor para que eu possa vê-los.

Para toda a parte geral do controle de versão do Git, consulte na 5. página de informações gerais (https://www.cs.columbia.edu/~du/cs/homework_Instructions.html)

ÿ Problemas Escritosÿ

1. Contraste TCP e UDP. Em que circunstâncias você escolheria um em detrimento do outro?

2. Qual é a diferença entre cache e replicação de dados?

3. Por que precisamos de um cache de nível de aplicativo, ou seja, por que os benefícios do cache de nível de aplicativo são sobre o nível de hardware ou do sistema operacional?

ÿ Problemas de programaçãoÿ

Perceber

Esta é a primeira tarefa da lição de casa da série yfs. Requer conhecimento básico de programação de rede Unix. Se você não for capaz de terminar esta lição de casa por conta própria, você deve reconsiderar seriamente fazer este curso.

Todas as atribuições de programação precisam ser escritas em C++.

Introdução

Neste trabalho de casa, você implementará um cliente TCP para recuperar arquivos e um servidor TCP para servir os arquivos solicitados. O cliente, após estabelecer uma conexão com o servidor, envia o nome do arquivo que solicita. Em seguida, o servidor procura o arquivo em seu cache de memória seguido por um diretório pré-definido. Se o arquivo for encontrado em qualquer um dos locais, o servidor então transmite o conteúdo do arquivo de volta ao cliente através da mesma conexão e armazena em cache o conteúdo do arquivo, se não estiver dentro do cache de memória.

A limitação de tamanho do cache de memória é de 64 MB. Se o tamanho do arquivo for maior que 64 MB no disco, você não o armazene em cache na memória. E se o tamanho do cache exceder 64 MB depois de colocar o arquivo na memória, você deve limpar o conteúdo em cache suficiente para garantir que todo o tamanho do cache seja restrito a 64 MB a qualquer momento.

Inclua um arquivo README junto com seu código. Você pode iniciar sua tarefa anotando nesse arquivo as seguintes informações:

- Uma visão geral que inclui a estrutura do seu programa Os detalhes da
- implementação do cache Diferentes casos de teste que você usa para
- testar seu programa: por exemplo, solicitar um arquivo inválido, solicitar de um servidor morto

Visão geral do dever de casa

O ÿuxo do programa deve se parecer com a seguinte descrição:

1. O lado do servidor cria um soquete, liga-o e escuta uma porta predefinida. Depois disso, ele bloqueia até que um cliente se conecte.

   ```
   % tcp_server port_to_listen_on file_directory
   por
   exemplo % ./tcp_server 9089 /home/dist/homework1
   ```

   O argumento acima fará com que o servidor escute na porta 9089 e procure o arquivo na `/home/dist/homework1 .`
2. O cliente se conecta ao servidor e envia uma solicitação para um arquivo. O lado do cliente recebe 4 parâmetros - o nome do servidor, a porta do servidor, o arquivo a ser solicitado e o diretório local para salvar o arquivo. Fica assim:

   ```
   % tcp_client server_host server_port file_name diretório
   por
   exemplo % ./tcp_client 59.78.58.28 9089 homework1.html .
   ```

   O argumento acima fará com que o cliente se conecte ao servidor 59.78.58.28:9089 e solicite o arquivo homework1.html e salve-o no diretório atual.
3. Assim que um cliente estiver conectado, o servidor analisará a solicitação e descobrirá qual arquivo o cliente está solicitando. Ele imprime uma linha

   `"Cliente xxx.xxx.xxx.xxx está solicitando o arquivo X"` , onde xxx.xxx.xxx.xxx é o cliente Endereço IP e X é o nome do arquivo que está sendo solicitado.
   - Se o arquivo estiver no cache, ele retornará o arquivo diretamente e exibirá `"Acerto de cache. Arquivo X enviado ao cliente"`. Caso contrário, ele encontrará o arquivo no diretório pré-definido e o salvará no cache e, em seguida, o retornará ao cliente e exibirá `"Cache miss. File X send to the client"` , onde X é o nome do arquivo.
   - Se o arquivo não existir, ele retornará uma mensagem de erro. Ele imprime uma linha

"Arquivo X não existe", onde X é o nome do arquivo. Observe que o servidor apenas procura o arquivo no diretório fornecido, não em outros diretórios.

Uma saída típica no lado do servidor se parece com isso:

```
% ./tcp_server 9089 /home/dist/homework1 O cliente 59.78.55.65 está
solicitando o arquivo homework1.html Cache miss. homework1.html enviado ao cliente

O cliente 59.78.55.66 está solicitando o arquivo homework1.html Cache hit. homework1.html
enviado ao cliente
O cliente 59.78.55.66 está solicitando o arquivo homework1.html Cache miss.
homework1.html enviado ao cliente
O cliente 59.78.55.65 está solicitando o arquivo homework1.html Cache hit. homework1.html
enviado ao cliente
```

[illegible] arquivo local, ou ele exibirá uma mensagem de erro se o arquivo não existir no o servidor. Nesse, caso, imprime uma linha

"Arquivo X não existe no servidor".

Uma saída típica no lado do cliente se parece com isso:

```
% /tcp_client 59.78.58.28 9089 homework0.html . homework0.html salvo
```

Você NÃO precisa implementar um servidor multi-thread - uma única versão de thread será boa.

Você é obrigado a lidar com o erro normalmente. Você precisará verificar o valor de retorno de cada função, como vincular e receber. Se houver algum erro, você imprimirá a mensagem de erro correspondente.

Dicas

- Você pode achar std::map (http://www.cplusplus.com/reference/stl/map/) útil ao implementar o cache. Observe que todos os arquivos solicitados estão no mesmo diretório, o que significa que todos eles têm nomes de arquivos diferentes, então a chave do mapa pode ser o nome do arquivo, e o valor seria o conteúdo do arquivo. Tenha muito cuidado com o std::map::operator[] (http://www.cplusplus.com/reference/stl/map/operator[]/), pois isso irá inserir um novo elemento no mapa se a chave não existe no mapa atual.
  Você pode simplesmente usar std::map::ÿnd (http://www.cplusplus.com/reference/stl/map /ÿnd/) em vez disso.
- Você precisará acompanhar o uso de memória do cache, pois exigimos que o cache nunca exceda 64 MB. Assim, você precisará implementar uma política de substituição de cache apropriada. O algoritmo LRU (menos usado recentemente) será eficiente, mas um algoritmo de substituição aleatória simples também está OK.

- Se você precisar de alguma atualização com a programação de soquete UNIX, você pode achar este guia (ftp://gaia.cs.umass.edu/cs653/sock.ps) útil. As seções 1 a 4 serão suficientes para este dever de casa.
- Não use strncpy (http://www.cplusplus.com/reference/clibrary/cstring/strncpy/) ao transferir dados do disco para o cache. Use memcpy (http://www.cplusplus.com/reference/clibrary/cstring/memcpy/) em vez disso. O strncpy (http://www.cplusplus.com/reference/clibrary/cstring/strncpy/) se comporta incorretamente ao lidar com arquivos binários.

Classificação

NOTA: Este primeiro dever de casa é classificado apenas como Aprovado/Reprovado

Vamos compilar e testar seu código no ambiente Ubuntu X86_64 LTS 12.04.2, ou seja, máquinas clic. A falha ao compilar em tal ambiente levará a um 0! Leia as instruções (https://www.cs.columbia.edu/~du/ds/homework-instructions.html) para obter mais informações sobre como testar seu programa tanto localmente quanto no ambiente clic

Seu programa será avaliado nos seguintes componentes:

1. Correção: Se o programa se comporta corretamente, por exemplo, ele transfere o arquivo em sua totalidade? ele armazena o conteúdo corretamente?
2. Estilo de codificação: seu código é legível com nomenclatura de variável adequada? Existem códigos confusos por aí? As chaves são consistentes?
3. Documentação: Existem comentários suficientes no código para torná-lo compreensível? O README contém documentação sucinta que descreve a sua solução?

Nota: Se algum de Erro de Segmentação for encontrado, você receberá metade das notas que poderia Tratamento de falhas: O programa emite uma mensagem informativa quando um erro 4 ocorre? ter obtido. E se você emitir "Não" para todas as mensagens de erro, vou apenas contá-las e deduzir a soma.

Certifique-se de ler as Políticas (https://www.cs.columbia.edu/~du/ds/policies.html) antes de começar. Você deve escrever todo o seu código sozinho. Usamos MOSS (http://theory.stanford.edu/~aiken/moss/) para plágio de software. Você pode experimentar você mesmo

ÿ

lição de casa 7 (https://www.cs.columbia.edu/~du/ds/categories.html#homework-ref) ÿ 6 de 7 lição de casa1 1 (https://www.cs.columbia.edu/~du/ ds/tags.html#homework1-ref)
dever de casa 7 (https://www.cs.columbia.edu/~du/ds/tags.html#homework-ref)

ÿ Arquivo anterior (https://www.cs.columbia.edu/~du/ds/archive.html)
Próximo ÿ (https://www.cs.columbia.edu/~du/ds/homework/2013/09/12/Homework2)