INSTITUTO FEDERAL
Paraná
Campus Pinhais

Bacharelado em ciência da computação
Sistemas de software livre — 22 de maio de 2025
Prova 1

Nome: Gabarito

Instruções

- A prova é individual e sem consulta.
- A interpretação do enunciado é parte da resposta.
- As respostas devem ser legíveis, completas, sucintas e objetivas.
- Não é permitido o uso de qualquer dispositivo eletrônico.
- As respostas devem ser escritas a caneta azul ou preta.
- Esta folha de prova deve ser devolvida. Você pode escrever respostas no verso.
- Após o início da prova, quem sair da sala não poderá retornar.
- Após o início da prova, se alguém sair, ninguém mais poderá entrar em sala.
- Qualquer violação dessas instruções pode acarretar no conceito D e nas devidas providências administrativas.
- Escreva seu nome em todas as folhas que entregar.
- Ao sair, assine a lista de presença.

1. Considere os comandos abaixo: ⟨20 pontos⟩

```
dir=~/shell
find "$dir" -type f -name "*a" | wc -l
```

(a) Qual a ideia geral dos comandos acima?
(b) Existem duas expansões nos comandos. Identifique e explique o que cada uma significa.
(c) Identifique os argumentos e opções de cada comando acima e diga o que eles fazem.
(d) Explique o que o *pipe* (`|`), na última linha, faz.
(e) Escreva uma árvore de diretórios que contenha 5 arquivos em `$dir`: 3 devem corresponder à expansão usada em `find` e 2 não devem corresponder.

2. Suponha que no diretório atual de trabalho existe um arquivo de texto chamado `hino.txt`. Escreva um comando para listar as linhas do arquivo que contêm a palavra "nosso". As linhas devem ser ordenadas inversamente. Você deve resolver o problema com um comando de uma linha usando *pipes*. Além de escrever o comando, você deve identificar e explicar todos os argumentos, opções, expansões e *pipes* que usar. ⟨20 pontos⟩

3. O programa `g++` é um compilador que lê arquivos C++ e os converte em um programa executável. Uma forma bem comum de invocar este programa é `g++ arquivo.cpp -o meuprograma`, que compila o `arquivo.cpp` (com código C++) em um arquivo executável que você pode usar (`meuprograma`). Você baixou todos os seus exercícios de Introdução à Programação no diretório `~/Downloads/intro_prog`. Sua tarefa é escrever um *script shell* que compile, um de cada vez, todos os arquivos que você baixou no diretório de `intro_prog`. Considere que você acabou de abrir um *shell* e que o diretório inicial de trabalho é `~`. ⟨20 pontos⟩

4. Escreva um *script* que faz o *backup* (cópia de segurança) das suas informações. Seu *script* deve receber como argumentos um diretório de origem e um diretório de destino, e copiar o conteúdo da origem no destino. Você deve verificar se os diretórios existem e criar o destino, se não existir. Além disso, você deve adicionar, no nome do destino, a data e hora em que a cópia foi feita. ⟨20 pontos⟩

(a) (Opcional) Altere seu *script* para que ele crie um arquivo compactado de *backup*, ao invés de apenas copiar o conteúdo da orgiem para o destino. O arquivo compactado ainda deve ser armazenado no diretório destino, e conter a informação de data e hora no nome.

5. Seu professor tem momentos deveras preguiçosos e, por isso, configurou a seguinte função no *shell* que utiliza para sincronizar os arquivos das disciplinas entre vários computadores. Sabendo que `git` é um programa que faz essa sincronia, responda as questões. ⟨20 pontos⟩

```
function gitclass {
 pushd ~/git
 echo "----------"
 echo "Executando git com o comando ${1}"
 echo;

 for proj in controle-disciplinas notas-de-aula webpage; do
  echo "==========";
  echo "Projeto: ${proj}";
  git -C ${proj} ${1}; # saída: ===== git ${proj} ${1} executou
  echo "==========";
  echo;
 done

 echo "Terminado: git ${1}"
 echo "----------"
 popd
}
```

(a) Explique o que a função faz.

(b) Escreva um exemplo de saída se a função for invocada assim: `gitclass pull`. Quando o comando `git` for executado, considere que ele gera a saída `===== git proj arg executou`, como escrito no comentário do código.

1) a) Encontrar todos os arquivos do diretório ~/shell, com nome terminado em "a" e contar quantos desses arquivos existem.

b) ~: diretório do usuário (home, inicial, principal)
"*a": expande para qualquer sequência de caracteres terminados em "a".

c) find: encontrar arquivos
"$dir": argumento, onde procurar por arquivos
-type f: opções para procurar por arquivos
-name: opção para filtrar arquivos com nome "*a"
wc: contar bytes, palavras e linhas
-l: contar linhas

d) O pipe conecta a saída do find à entrada de wc, de forma que wc vai trabalhar sobre o que o comando find produzir na saída.

e) ~
- shell
  - alemanha
  - belgica
  - brasil
  - finlandia
  - belize

2) - cat hino.txt | grep nosso | sort -r
- grep nosso hino.txt | sort -r

- cat hino.txt : lê conteúdo do arquivo
- pipe encaminha conteúdo do arquivo para grep
- grep filtra a entrada e imprime apenas linhas contendo a palavra "nosso".
- pipe encaminha linhas com "nosso" para o sort
- sort ordena as linhas lexicograficamente
  - -r aplica a ordem reversa.

3)
```
#!/bin/bash

cd ~/Downloads/intro_prog
for arquivo in $(ls); do
    g++ $arquivo -o $arquivo.out
done
```

4) #!/bin/bash

```
src=$1
dst=$2

if [ ! -d $src ]; then
    echo "Origem não existe"
    exit 1
fi

mkdir -p $dst/$src-$(date)
cp -r $src $dst/$src-$(date)
```

Opcional

```
mkdir -p $dst
tar czvf $dst/$src-$(date) $src
```

5)

a) pushd ~/git : vai para ~/git, salvando o diretório anterior

Depois, para cada projeto (controle-disciplinas, notas-de-aula, webpage), usa o git para sincronizar os arquivos.

b) ----------

Executando git com o comando pull

==========

Projeto: controle-de-disciplinas

===== git controle-de-disciplinas pull executou

Projeto: notas-de-aula

git notas-de-aula pull executou

Projeto: webpage

git webpage pull executou

Terminado: git pull