Prática de Laboratório – Comandos Linux

O objetivo desta aula é apresentar comandos mais elaborados do Linux, via terminal de comandos, utilizando um conjunto de arquivos de exemplo (*sample*). A apresentação dos comandos tentará seguirá um ordem linear de manipulação dos arquivos. Abra um terminal de comandos e posicione-se dentro do diretório *sample* de nossa última aula.

1 Um comando que costuma ser muito útil para saber sobre quantidades de texto de um arquivo é o comando *wc (word count)*. Ele retorna a quantidade símbolos de 'nova linha' (*\n*), a quantidade de caracteres e de bytes presentes em cada arquivo ou entrada. Acione o comando *wc* sobre o arquivo *sequence.fasta*.

1.1 *wc sequence.fasta*

1.2 Compare o resultado acima com o retornado pelo comando: *cat sequence.fasta*

2 O comando *wc* consegue abrir arquivos por conta própria, mas pode ser que não se queira abrir um arquivo inteiro, mas apenas parte dele. Por exemplo, suponha que você queira ver as estatísticas apenas sobre a primeira linha do arquivo *sequence.fasta*. Então podemos usar os comandos head em um pipeline com o wc. Veja o resultado deste comando:

2.1 head -n 1 sequence.fasta | wc

2.2 Agora digite apenas *head -n 1 sequence.fasta* e compare com o que foi retornado pelo pipeline.

2.3 Qualquer formato textual gerado pelos comandos pode ser direcionada como entrada para o processamento do comando *wc*. Por exemplo, como saber quantos arquivos existem dentro do diretório *proteinas*? (sem acento) Vamos lá, tente, na pior das hipóteses você pode baixar os arquivos novamente e recomeçar!

3 O comando grep é bastante útil porque nos permite a consulta por expressões regulares em vários arquivos. Vamos, por exemplo, considerar o exemplo de sala de aula: *Um determinado motivo proteico conservado AKAP foi descrito na literatura como importante para a instauração de uma infeção bacteriana. As proteínas do seu organismo de estudo possuem este motivo?* Para resolver esse problema basta digitar o comando abaixo e analisar o resultado:

3.1 *grep AKAP proteinas/**

3.2 Entenda como o resultado foi retornado, esse entendimento e importante para saber como usar corretamente o comando em conjunto com outros.

4 O comando *grep* também possibilita fazer a contagem da quantidade de resultados retornados. Ao invés de você ficar contando na tela a quantidade de motivos retornados (destacados em vermelho) é muito mais prático contá-los com o uso do próprio *grep*. Acesse o manual do *grep* e descubra qual parâmetro o configura para fazer a contagem, ao invés de listar o motivo buscado na tela. Em seguida:

4.1 Use este parâmetro no último comando linux para contar quantos motivos existem.

5 Agora façamos a busca no arquivo mt2.fasta por um motivo/padrão que é apenas o sinal de maior (>). Porquê? É simples: o sinal de maior em um arquivo de proteínas ou nucleotídeos sempre indica o início de um cabeçalho - a parte que dá nome - a uma sequencia de aminoácidos (proteína) ou nucleotídeos (DNA). Dessa forma se contarmos quantos sinais de maior existem em um arquivo fasta significa que estamos contando quantas sequencias existem em um arquivo multi-fasta (um fasta com várias sequencias).

5.1 Posicione sua janela de comandos no diretório onde está o seu arquivo *mt.fasta* ou *mt2.fasta*. Se achar conveniente pode criar o arquivo *mt.fasta* novamente, lembra-se de como fazê-lo? Uma dica: *cat* isso > aquilo.

5.2 O comando *grep '>' mt.fasta* apenas lista todas as linhas que possuem o sinal de maior. Verifique executando este comando. Perceba que não é isso o que queremos. Dessa forma utilize o parâmetro que você descobriu no item anterior para contar quantas sequencias existem dentro do arquivo. Compare o resultado com o retornado pelo item *2.3*.

6 O comando *find* nos permite vasculhar todo um ambiente computacional de maneiras inimagináveis e até mesmo impossíveis se comparado com as opções oferecidas pelos sistemas baseados apenas em janelas. Nenhum arquivo ficará escondido de você, nunca mais, desde que você se habitue a utilizar este comando. Ah, funciona também com arquivo dos sistemas de arquivos FAT (Windows): caso você tenha o Linux compartilhando fisicamente a máquina com o Windows (sem usar máquina virtual) você consegue mapear o disco do Windows e navegar por ele com uma janela de comandos Linux. Não faremos isso nesta aula de hoje, mas fiquem à vontade para tentarem em suas máquinas. Façamos algumas consultas com o find:

6.1 Encontre os arquivos modificados a mais de dois dias dentro do diretório sample com o comando *find . -mtime +2*

6.2 Encontre os arquivos modificados a menos de um dia dentro do diretório sample com o comando *find . -mtime -1*

6.3 Encontre os arquivos filhos que possuem um tamanho maior que 500 kilobytes com o comando *find . -size +500k*

6.4 Encontre os arquivos filhos que possuem um tamanho maior que 500 kilobytes e liste as propriedades básicas destes arquivos com o comando *find . -size +500k -ls*

6.5 Encontre todos os arquivos cujo nome termine com pdf com o comando *find . -name '*.pdf' -ls*

6.6 Encontre arquivos ou diretórios, a partir do diretório *sample*, que possuem a permissão de execução para o grupo 'outros'. Lembre-se que existem os grupos de permissões: usuário proprietário, grupo proprietário e outros, que utilizam as letras 'u', 'g' e 'o' para significar cada grupo, respectivamente, sendo que as permissões podem ser r=read, w=write e x=executar. Este comando gera o relatório: *find . -perm -o=x*

6.7 Encontre os diretórios a partir da pasta *sample* que possuem permissão de escrita para o grupo proprietário: *find . -perm -g=w -type d*

6.8 Encontre e liste os arquivos de proteínas, da pasta *proteinas*, apenas com arquivos cujas sequencias de proteínas forem maiores do que 1 kilobyte (1k): *find ./proteinas -size +1k -type f -ls*

6.9 Crie um arquivo multi-fasta denominado *maiores.fasta*, a partir da pasta *proteinas*, contendo apenas os arquivos com sequencias de proteínas que forem maiores do que 1 kilobyte (1k): *find ./proteinas -size +1k -type f -exec cat > maiores.fasta {} \;*

6.10 Encontre os mesmos arquivos do item anterior, mas ao invés criar um arquivo com as sequencias, execute um comando grep para procurar pelo motivo conservado AKAP: *find ./proteinas -size +1k -type f -ls -exec grep AKAP {} \;*

7 O comando *sed* nos auxilia realizando edições em lote em diversos arquivos. Vejamos um exemplo simples de uso do *sed*, os exemplos mais interessantes são criados quando conjugamos o *sed* com expressões regulares. Suponha que uma novo motivo proteico está sendo estudado, o motivo *ACP*. Deseja-se editar o arquivo *maiores.fasta*, criado nos item anterior, para delimitar todas as ocorrências de ACP com um tipo de *tag,* de modo a que este motivo não fique oculto em possíveis análises visuais do arquivo. Para tanto onde houver "ACP" deve ser substituido por "< ACP >". Execute este comando para realizar tal operação: *sed -i "s/ACP/< ACP >/g" maiores.fasta*

8 Vimos muitos exemplos de Expressões Regulares (ER's) em sala de aula. O comando *man grep* possui um manual completo sobre ER's, recomendo a leitura em outra oportunidade.

8.1 Os comandos grep e sed se tornam muito poderosos quando expressões regulares são utilizadas nos parâmetros de busca. Um exemplo simples pode ser a busca de um motivo conservado no qual existe um aminoácido que pode variar entre n letras, digamos n=3 letras. Suponha o motivo AKAP no qual a segunda letra A pode ser substituída pelos aminoácidos R e P. Este comando gera o relatório que precisamos:
*grep "AK[ARP]P" proteinas/**

8.2 Outra variação do comando anterior seria a possibilidade de que qualquer outro aminoácido pudesse entrar no lugar da segunda letra A. Neste caso o ponto '.' vem ao nosso favor: *grep "AK.P" proteinas/**

8.3 Mais uma variação do comando inicial. Suponha que um outro motivo possua uma ou mais letras A no local onde ocorre a segunda letra A. Neste caso o operador de repetição de ER's resolve a nossa consulta. Nessa caso o operador '+' precisa ser precedido por uma contra-barra: *grep "AKA\+P" proteinas/**

8.4 Uma variação do último comando. Suponha que no lugar do segundo A possa existir zero ou mais ocorrências de A: *grep "AKA*P" proteinas/**

9 Quando o comando sed utiliza ER's podemos evitar a escrita de pequenos programas para os quais não há opções desenvolvidas que servem a necessidades urgentes e as vezes únicas. Um exemplo: para que os arquivos com as proteínas de uma bactéria sejam reconhecidas por um determinado programa eles precisam incorporar no início do nome, por exemplo, o prefixo *Mt* que representa a abreviatura do organismo (*Mycobacterium tuberculosis*).

9.1 Liste os cabeçalhos dos arquivos presentes no diretório proteinas com o comando: *grep ">" proteinas/**

9.2 Modifique o cabeçalho de todos os arquivos dentro da pasta *proteinas* para incluir o prefixo Mt assim adaptar os arquivos para utilização com o tal programa:
*sed -i "s/>\(Rv[0-9]\+c\?\)/>Mt_\1/g" proteinas/**

9.3 Execute o passo 9.1 novamente.