2012 - BMMB 597D: Analyzing Next Generation Sequencing Data

Week 2, Lecture 4

István Albert

Biochemistry and Molecular Biology and Bioinformatics Consulting Center

Penn State
Shell scripts

Collect multiple commands into a single program

• Run the same commands again or on other data

• Document the steps and describe the thought process
Creating and refining a shell script

```bash
# a shell script is a text file with commands

cat features.gff | awk -f counter.awk

ialbert@porthos ~/work/lec4
$ ls
counter.awk coverage.sh features.gff

ialbert@porthos ~/work/lec4
$ sh coverage.sh
40586145

ialbert@porthos ~/work/lec4
$ 
```
Add another step

# you can add multiple steps
```
cat features.gff | awk '$3 == "gene" { print $0 }' > tmp
```

# you may build a long chain or save intermediate results
```
cat tmp | awk -f counter.awk
```

```
IALIZBERT@PORTHOS ~/work/lec4
$ ls
counter.awk coverage.sh features.gff tmp

IALIZBERT@PORTHOS ~/work/lec4
$ sh coverage.sh
8886525

IALIZBERT@PORTHOS ~/work/lec4
$ 
```
Using Shell Variables

```bash
# parametrize the script
FILE=features.gff
PROG=counter.awk

# generate information
echo "Running $PROG on $FILE"

cat $FILE |

# you may build
ialbert@porthos ~/work/lec4

$ ls
counter.awk coverage.sh features.gff tmp

ialbert@porthos ~/work/lec4
$ sh coverage.sh
Running counter.awk on features.gff
8886525

ialbert@porthos ~/work/lec4
$ 
```
Single and double quoted strings

```bash
# parametrize the script
FILE=features.gff
PROG=counter.awk

difference between single and double quotes
echo "Running $PROG on $FILE"
echo 'Running $PROG on $FILE'

# you can add more arguments
cat $FILE awk

# you may build
cat tmp awk -f
```
• Remember that you may be using distinct languages – shell, awk, perl, python variables may look the same but are interpreted differently!

• Avoid mixing the context -  i.e. awk program modified via bash variables. Instead write the awk program separately and pass the variables into it.

```bash
# you can add multiple steps
cat $FILE | awk '$3 == "gene" { print $0 }' > tmp
```
# parametrize the script

FILE=features.gff
PROG=counter.awk

# what happens if you make mistakes
echo "Running $PROG on $FILE"
echo "Running $PROG on $LILE"

^C

ialbert@porthos ~/work/lec4
$ sh coverage.sh
Running counter.awk on features.gff
Running counter.awk on
coverage.sh: line 10: program1: command not found
coverage.sh: line 11: program2: command not found

ialbert@porthos ~/work/lec4
$
Strict error checking

```bash
# parametrize the script

# strict error checking mode
set -ue

FILE=features.gff
PROG=counter.awk

# what happens if you make mistakes
echo "Running counter.awk on features.gff"
coverage.sh: line 10: LILE: unbound variable

ialbert@porthos ~/work/lec4
$ sh coverage.sh
Running counter.awk on features.gff
coverage.sh: line 12: programl: command not found

ialbert@porthos ~/work/lec4
$ sh coverage.sh
```

```bash
```
```bash

ialbert@porthos ~/work/lec4
```
Looping over multiple files

```bash
# useful constructs
# strict error checking mode
set -ue
PROG=counter.awk

# there are subtle rules about shell file name expansion
for FILE in *.gff
do
  echo "Processing $FILE"
  cat $FILE | awk '{3 == "gene" { print $0 }' > tmp
  cat tmp | awk -f $PROG
done
```

```
ialbert@porthos ~/work/lec4
$ cp features.gff data1.gff

ialbert@porthos ~/work/lec4
$ sh coverage.sh
Processing data1.gff
8886525
Processing features.gff
8886525

ialbert@porthos ~/work/lec4
$ 
```
Bash has lots of features

We will slowly introduce some features along the way
Entrez Programming Utilities: EUtils

• Query and download Entrez (Genbank and other databases) via URLs

• Combined with UNIX tools allow you to automatically download data

• Named as efetch, esearch, etc.
Download via Eutils

```bash
# strict error checking mode
set -ue

# every url has this start

# fetch url
FETCH="efetch.fcgi?db=protein&id=15718680"

URL="$BASE$FETCH"

echo "downloading $URL"
curl -S $URL > data.txt

ialbert@portos ~/work/lec4
$ sh getdata.sh
```
Refine our script

```bash
# every url has this start

# database
DB="protein"

# one or more ids
IDS=15718680,157427902,119703751

# the encoding text, html or xml
MODE=text

# the file format fasta, genbank, asn
TYPE=fasta

# fetch url
FETCH="efetch.fcgi?db=$DB&id=$IDS&retmode=$MODE&rettype=$TYPE"
```
## Valid databases

<table>
<thead>
<tr>
<th>Entrez Database</th>
<th>UID common name</th>
<th>E-utility Database Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>BioProject</td>
<td>BioProject ID</td>
<td>bioproject</td>
</tr>
<tr>
<td>BioSample</td>
<td>BioSample ID</td>
<td>biosample</td>
</tr>
<tr>
<td>Biosystems</td>
<td>BSID</td>
<td>biosystems</td>
</tr>
<tr>
<td>Books</td>
<td>Book ID</td>
<td>books</td>
</tr>
<tr>
<td>Conserved Domains</td>
<td>PSSM-ID</td>
<td>cdd</td>
</tr>
<tr>
<td>dbGaP</td>
<td>dbGaP ID</td>
<td>gap</td>
</tr>
<tr>
<td>dbVar</td>
<td>dbVar ID</td>
<td>dbvar</td>
</tr>
<tr>
<td>Epigenomics</td>
<td>Epigenomics ID</td>
<td>epigenomics</td>
</tr>
<tr>
<td>EST</td>
<td>GI number</td>
<td>nucest</td>
</tr>
<tr>
<td>Gene</td>
<td>Gene ID</td>
<td>gene</td>
</tr>
<tr>
<td>Genome</td>
<td>Genome ID</td>
<td>genome</td>
</tr>
<tr>
<td>GEO Datasets</td>
<td>GDS ID</td>
<td>gds</td>
</tr>
<tr>
<td>GEO Profiles</td>
<td>GEO ID</td>
<td>geoprofiles</td>
</tr>
<tr>
<td>GSS</td>
<td>GI number</td>
<td>nucgss</td>
</tr>
<tr>
<td>HomoloGene</td>
<td>HomoloGene ID</td>
<td>homologene</td>
</tr>
<tr>
<td>MeSH</td>
<td>MeSH ID</td>
<td>mesh</td>
</tr>
<tr>
<td>NCBI C++ Toolkit</td>
<td>Toolkit ID</td>
<td>toolkit</td>
</tr>
<tr>
<td>NCBI Web Site</td>
<td>Web Site ID</td>
<td>ncbisearch</td>
</tr>
<tr>
<td>NLM Catalog</td>
<td>NLM Catalog ID</td>
<td>nlmcatalog</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>GI number</td>
<td>nuccore</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>GI number</td>
<td>nuccore</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>E-utility Database Name</th>
<th>Entrez Database</th>
<th>UID common name</th>
</tr>
</thead>
<tbody>
<tr>
<td>nucore</td>
<td>Nucleotide</td>
<td>GI number</td>
</tr>
<tr>
<td>omia</td>
<td>OMIA</td>
<td>OMIA ID</td>
</tr>
<tr>
<td>popset</td>
<td>PopSet</td>
<td>PopSet ID</td>
</tr>
<tr>
<td>probe</td>
<td>Probe</td>
<td>Probe ID</td>
</tr>
<tr>
<td>protein</td>
<td>Protein</td>
<td>GI number</td>
</tr>
<tr>
<td>proteinclusters</td>
<td>Protein Clusters</td>
<td>Protein Cluster ID</td>
</tr>
<tr>
<td>pcassay</td>
<td>PubChem BioAssay</td>
<td>AID</td>
</tr>
<tr>
<td>pccompound</td>
<td>PubChem Compound</td>
<td>CID</td>
</tr>
<tr>
<td>pcsubstance</td>
<td>PubChem Substance</td>
<td>SID</td>
</tr>
<tr>
<td>pubmed</td>
<td>PubMed</td>
<td>PMID</td>
</tr>
<tr>
<td>pmc</td>
<td>PubMed Central</td>
<td>PMCID</td>
</tr>
<tr>
<td>rsn</td>
<td>SNP</td>
<td>rs number</td>
</tr>
<tr>
<td>sra</td>
<td>SRA</td>
<td>SRA ID</td>
</tr>
<tr>
<td>structure</td>
<td>Structure</td>
<td>MMDB-ID</td>
</tr>
<tr>
<td>taxonomy</td>
<td>Taxonomy</td>
<td>TaxID</td>
</tr>
<tr>
<td>unigene</td>
<td>UniGene</td>
<td>UniGene Cluster ID</td>
</tr>
<tr>
<td>unists</td>
<td>UniSTS</td>
<td>STS ID</td>
</tr>
</tbody>
</table>
Valid return types and modes

<table>
<thead>
<tr>
<th>db = nuccore, nucest, nucgss, protein or popset</th>
</tr>
</thead>
<tbody>
<tr>
<td>text ASN.1</td>
</tr>
<tr>
<td>binary ASN.1</td>
</tr>
<tr>
<td>Full record in XML</td>
</tr>
<tr>
<td>Accession number(s)</td>
</tr>
<tr>
<td>FASTA</td>
</tr>
<tr>
<td>TinySeq XML</td>
</tr>
<tr>
<td>SeqID string</td>
</tr>
</tbody>
</table>

**Additional options for db = nuccore, nucest, nucgss or popset**

| GenBank flat file                             | gb | text |
| GBSeq XML                                     | gb | xml |
| INSDSeq XML                                   | gbc | xml |

**Additional option for db = nuccore and protein**

| Feature table                                 | ft | text |
Adding more interactivity

```bash
set -ue

# every url has this start

# database, specified as command line parameter 1
DB=$1

# one or more ids, specified as command line parameter 2
IDS=$2

# the encoding text, html or xml
MODE=text

# the file format fasta, genbank, asn
TYPE=gb

# fetch url
wget $BASE$DB$IDS/MODE$TYPE
```

```
imbert@porthos ~/work/lec4
$ sh getdata.sh protein 15718680
```
Always investigate files

```
ialbert@porthos ~/work/lec4
$ cat data.gb.text | grep -A 1 Region | head
  Region       6..109
  /region_name="PH_Tec"
  Region      113..148
  /region_name="BTK"
  Region      174..229
  /region_name="SH3_ITK"
  Region      232..340

ialbert@porthos ~/work/lec4
$ 
```
XML is a file markup format but it is not a data format.
Advanced topic: XSLT transformations

• An XML document can be transformed by rules described in another XML document

• XSL Transformations

• You don’t need to know these, but are very handy if someone can make one for you

• Programmers love to make these post it on http://www.stackoverflow.com
Example XSLT

```xml
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text" omit-xml-declaration="yes" indent="no"/>
  <xsl:template match="/">
    <xsl:for-each select="/GBFeature">
      <xsl:value-of select="GBFeature_key/text()"/>
      <xsl:text> </xsl:text>
      <xsl:value-of select="GBFeature_location/text()"/>
      <xsl:text>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

```bash
$ xsltproc format.xml data.xml | head -5
source 1..620
Protein 1..620
Region 6..109
Site order(13,15..16,18..19,25,29,40,52)
Region 113..148
```

ialbert@porthos ~/work/lec4
$
Homework 4

• Create a shell script that can download three proteins of interest to you from the protein database.

• Write a shell script that lists the journal papers that GeneBank references for your data