Array and hash are VERY useful to hold text data in the memory and are often created using loops

- Creating Array:

@fruit_list = ('apple', 'orange', 'banana'); =

```
$fruit_list[0]='apple';
$fruit_list[1]='orange';
$fruit_list[2]='banana';
```

@a=<>; ←

```
apple
orange
banana
```

- Creating Hash:

```
%ip2hostname = (
"glu" => "131.156.41.196",
"gly" => "131.156.41.193",
"cys" => "131.156.41.195"
);
```

=

```
$ip2hostname{"glu"}='131.156.41.196';
$ip2hostname{"gly"}='131.156.41.193';
$ip2hostname{"cys"}='131.156.41.195';
```

? ←

glu	131.156.41.196
gly	131.156.41.193
cys	131.156.41.195

```
#!/usr/bin/perl

open (IN,$ARGV[0]);

@a=<IN>;

foreach(@a){
    @col=split(/\t/,$_);
    print $col[1],"\tmutation\n";
}
```

=

```
#!/usr/bin/perl

while (<>){
    @col=split(/\t/,$_);
    print $col[1],"\tmutation\n";
}
```

There's More Than One Way To Do It

<u>The pro of while:</u>
No need to load all data into memory; process data on a line by line basis

<u>The con of while:</u>
Can not reference other lines and can only work once

= `cat cosmicRaw.txt.head10.6col | cut -f2 | awk '{print $1,"mutation"}' | sed 's/ /\t/'`

```
perl  array-from-file3.pl /home/yyin/work/class/cosmicRaw.txt.head10.6col
```

hash: name-value pair, name also called key

$hash{'key'}='value';

Like array, key and value of a hash can be assigned in a loop

Key and value could have strict one-to-one correspondence or not

COSMIC v63	COSM383711	39340	ENST00000401030	Substitution - Nonsense 1	lung	
COSMIC v63	COSM568223	A1BG	ENST00000263100	Substitution - coding silent	19	lung
COSMIC v63	COSM226401	A1BG	ENST00000263100	Substitution - Missense 19	NS	
COSMIC v63	COSM568222	A1BG	ENST00000263100	Unknown 19	lung	
COSMIC v63	COSM395741	A1BG	ENST00000263100	Substitution - Missense 19	lung	
COSMIC v63	COSM568221	A1BG	ENST00000263100	Substitution - coding silent	19	lung
COSMIC v63	COSM1002703	A1BG	ENST00000263100	Substitution - coding silent	19	endometrium
COSMIC v63	COSM1129683	A1BG	ENST00000263100	Substitution - Missense 19	prostate	
COSMIC v63	COSM339965	A1BG	ENST00000263100	Substitution - Missense 19	lung	
COSMIC v63	COSM308725	A1BG	ENST00000263100	Substitution - coding silent	19	lung

```
#!/usr/bin/perl -w

while(<>){
  @col=split(/\t/,$_);
  $cosmic{$col[1]}=$_;
}
print $cosmic{'COSM339965'};
```

Key: $col[1], the second col
Value: $_, the entire row

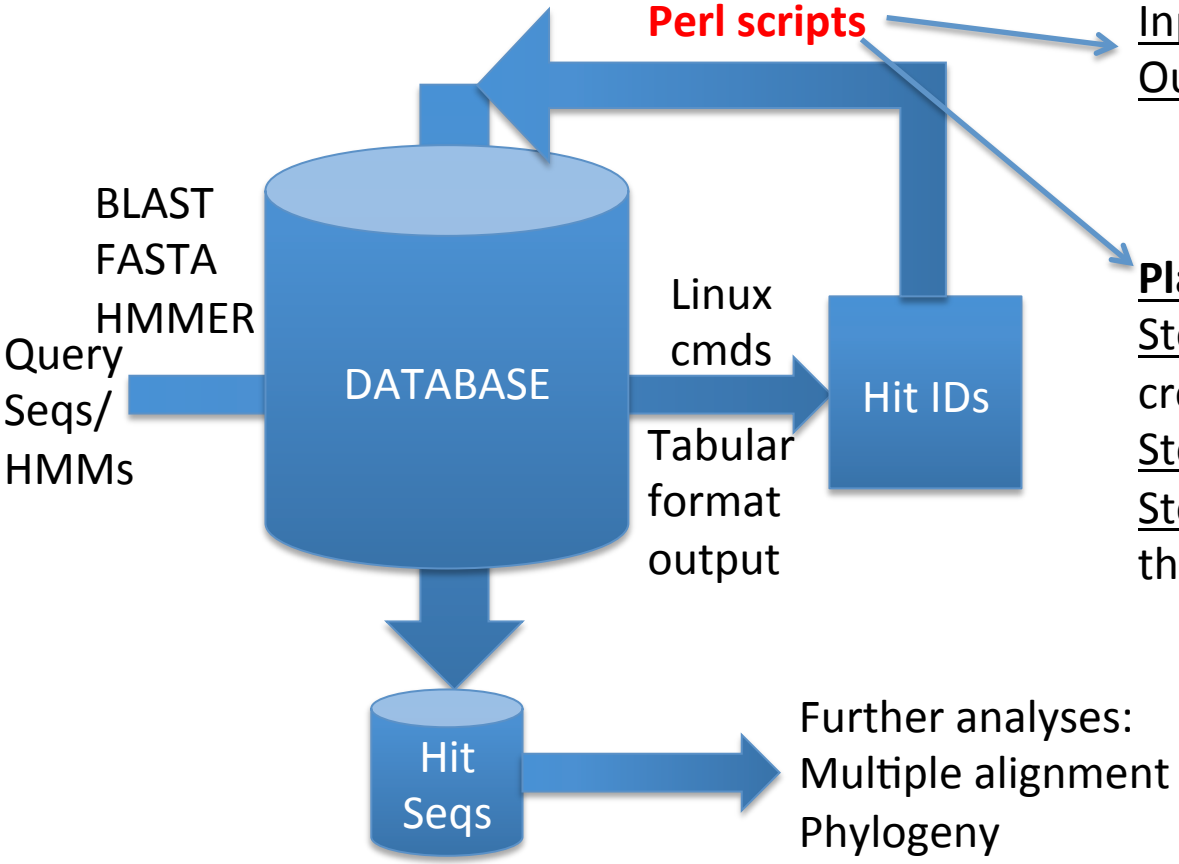
Call this particular key

```
perl hash-from-file.pl /home/yyin/work/class/cosmicRaw.txt.head10.6col
```

This could be VERY useful for holding fasta sequences: id as the key and sequence as the value

We can store a large fasta file in the memory and call any specific ids anytime

e.g. read in a list of subset IDs to extract the sequences

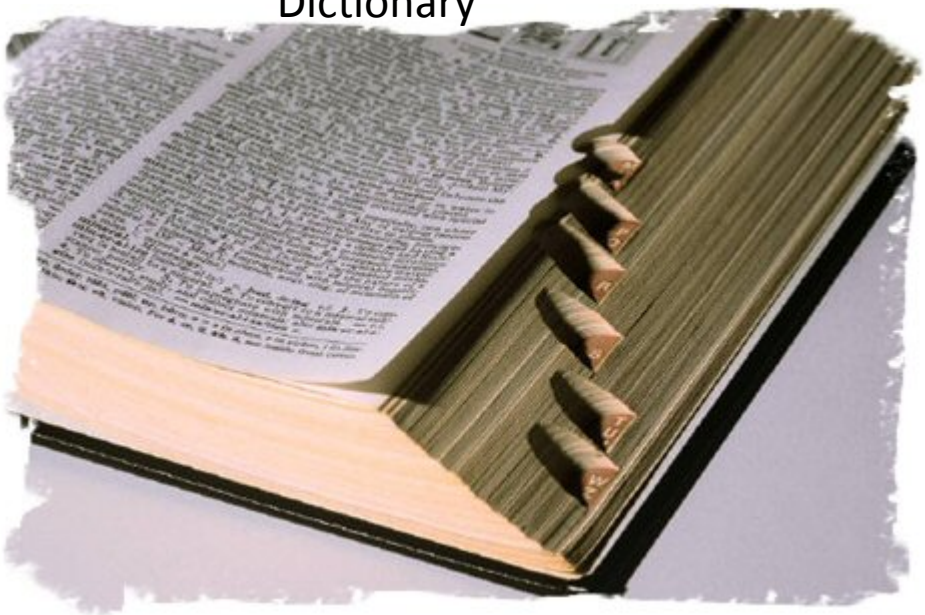


Key: a code for the food
Value: the food itself

Key: a sequence ID
Value: the sequence itself

Key: a word
Value: the detailed info/meaning of that word

Dictionary



Vending machine

do the following in /media/DATAPART1/your id/class/mar19/

```
[Now try something big (and slow)]
time blastall -p blastp -i test-query.fa -d /home/yyin/work/class/
metagenemark_predictions.faa -m 9 -e 1e-2 > test-
query.fa.cowrumen.out.m9 &

[Do some parsing]
less test-query.fa.cowrument.out.m9 | cut -f1,2,3,7- | less
less test-query.fa.cowrument.out.m9 | cut -f1,2,3,7- | grep -v '^#' |
cut -f2 | sort -u | head

# Save a test id file
less test-query.fa.cowrument.out.m9 | cut -f1,2,3,7- | grep -v '^#' |
cut -f2 | sort -u > test-query.fa.cowrument.out.m9.hitid
```

The database file:
/home/yyin/work/class/metagenemark_predictions.faa

The subset id file:
test-query.fa.cowrument.out.m9.hitid

As the first step, we need to load the /home/yyin/work/class/metagenemark_predictions.faa into memory

```
# But let's check a smaller fasta seq file first
less test-query.fa.cowrument.out.m9.head10.fa
```

```
>gi|388476126|ref|YP_488309.1| homoserine kinase
[Escherichia coli str. K-12 substr. W3110]
MVKVYAPASSANMSVGFDVLGAAVTPVDGALLGDVVTVEAAETFSLNNLGRFADKL
PSEPRENIVYQCWE
RFCQELGKQIPVAMTLEKNMPIGSGLGSSACSVVAALMAMNEHCGKPLNDTRLLAL
MGELEGRISGSIHY
DNVAPCFLGGMQLMIEENDIISQQVPGFDEWLWVLAYPGIKVSTAEARAILPAQYR
RQDCIAHGRHLAGF
IHACYSRQPELAAKLMKDVIAEPYRERLLPGFRQARQAVAEIGAVASGISGSGPTL
FALCDKPETAQRVA
DWLGKNYLQNQEGFVHICRLDTAGARVLEN
>NODE_3573325_length_256363_cov_8.500103_orf_03140
complement(202076..203146)
MKKIKVFAPASIANLGCGFDIMGMALDEVGDVLEMSLDEDSSGISIVNETDVPLPE
DIDQNVITPVIRKFFEMTGHSGRVDVRVLKKIYPGSGIGSSAASSAAAAFGINELF
GAPLSEEDVVVCAMEGENLASGGYHADNAAPAVMGGIILIRGYEPLDVIKLPVPGN
LYCPVIHPHLMVSTKAARSILPKEIPMHTAITQWGNVGGLVAGLCTGNIELVGRAM
RDAVAEPYRKGFIPGFDELRAKLLGAGALAMNISGSGPSVFALANRGDIAQRVGAI
MERHFAQQGILSETYVVKVXXXAAPQARGGRQALALRRQDGGRKVFRLPAGRREPA
FLLCTGRXXXXSNKGARLIA*
>NODE_3641687_length_21494_cov_1.063320_orf_137760
complement(19693..20634)
MKVSVRVPATVANIGPGFDCLGMALPIYNTITIEETVLPGTGIEINVLANEDVTDE
LSLEHIPMDENSIIYKAVELLYNSIGQTPSELKITIHSEIPIAKGLGSSASVIVGG
LIAANELLGKPADEAALLSIATEVEGHPDNITPAIIGGLTLSSAEEDGSIVSRNLP
WPEEWVLTVCVPEYELATEISRSVLPKEVPLTDAVYNAQRMAMFVQAIYTKDEELM
KLALRDKLHQPYRMKLVPGFDKISENLKHEESVLGVVLSGAGPSILVVSLKTNLDK
VKTIIKETWDELSINAQMYTLPIDKTGAVVIPE*
```

Description line does not allow to break into multiple lines (have multi- newline character)

Sometimes there are multiple lines (newline)

Step 1: process fasta seq file to create a hash (load into memory)

We want to get ride of newline character inside the sequences

vi get-seq.pl

Here is where the second entry started

The tabular space

```
#!/usr/bin/perl -w

while(<>){
  chomp $_;  # get ride of newline character
  if($_=~/>/){  # =~ is used to match regexp
    $_=~s/>//;  # =~s is used to substitute
    print $_,"\t"; # insert a tabular space
  }
  else{
    print $_; # print the sequence line (no
              # newline character anymore)
  }
}
```

if/else statement to control the flow

```
if (condition)
{
  action;
}
elsif
(condition){
  action;
}
else{
  action;
}
```

perl get-seq.pl test-query.fa.cowrument.out.m9.head10.fa

```
gi|388476126|ref|YP_488309.1| homoserine kinase [Escherichia coli str. K-12 substr. W3110]  MVKVYAPASSANMSVGFDVLGAAVT
PVDGALLGDVVTVEAAETFSLNNLGRFADKLPSEPRENIVYQCWERFCQELGKQIPVAMTLEKNMPIGSGLGSSACSVVAALMAMNEHCGKPLNDTRLLALMGELEGRISGSIHYDNVAPC
FLGGMQLMIEENDIISQQVPGFDEWLWVLAYPGIKVSTAEARAILPAQYRRQDCIAHGRHLAGFIHACYSRQPELAAKLMKDVIAEPYRERLLPGFRQARQAVAEIGAVASGISGSGPTLF
ALCDKPETAQRVADWLGKNYLQNQEGFVHICRLDTAGARVLENNODE_3573325_length_256363_cov_8.500103_orf_03140 complement(202076..203146)
      MKKIKVFAPASIANLGCGFDIMGMALDEVGDVLEMSLDEDSSGISIVNETDVPLPEDIDQNVITPVIRKFFEMTGHSGRVDVRVLKKIYPGSGIGSSAASSAAAAFGINELFGAP
LSEEDVVVCAMEGENLASGGYHADNAAPAVMGGIILIRGYEPLDVIKLPVPGNLYCPVIHPHLMVSTKAARSILPKEIPMHTAITQWGNVGGLVAGLCTGNIELVGRAMRDAVAEPYRKGF
IPGFDELRAKLLGAGALAMNISGSGPSVFALANRGDIAQRVGAIMERHFAQQGILSETYVVKVXXXAAPQARGGRQALALRRQDGGRKVFRLPAGRREPAFLLCTGRXXXXSNKGARLIA*
NODE_3641687_length_21494_cov_1.063320_orf_137760 complement(19693..20634) MKVSVRVPATVANIGPGFDCLGMALPIYNTITIEETVLPGTGIEIN
VLANEDVTDELSLEHIPMDENSIIYKAVELLYNSIGQTPSELKITIHSEIPIAKGLGSSASVIVGGLIAANELLGKPADEAALLSIATEVEGHPDNITPAIIGGLTLSSAEEDGSIVSRNL
PWPEEWVLTVCVPEYELATEISRSVLPKEVPLTDAVYNAQRMAMFVQAIYTKDEELMKLALRDKLHQPYRMKLVPGFDKISENLKHEESVLGVVLSGAGPSILVVSLKTNLDKVKTIIKET
WDELSINAQMYTLPIDKTGAVVIPE*NODE_3701631_length_68488_cov_2.143266_orf_51600 complement(15723..16385)  MGVSAFAPASIGNVSVG
FDILGAALKPIDGQILGDNVDVYAGDSEFDLSIEGWFASKLPADPKKNICYDAYVGFKALLEEKGIAVKPVKMVLKKNLPIGSGLGSSAASIVAAVEALNAFHDYPLSKDEALTLMGRLEG
```

```
#!/usr/bin/perl -w

while(<>){
   chomp $_;   # get ride of newline character
   if($_=~/>/){   # =~ is used to match regexp
     $_=~s/>//;   # =~s is used to substitute
     print "\n",$_,"\t"; # insert a tabular space
   }
   else{
     print $_; # print the sequence line (no
               # newline character anymore)
   }
}
```

We have an empty line in the beginning of the output but no newline in the end of the file

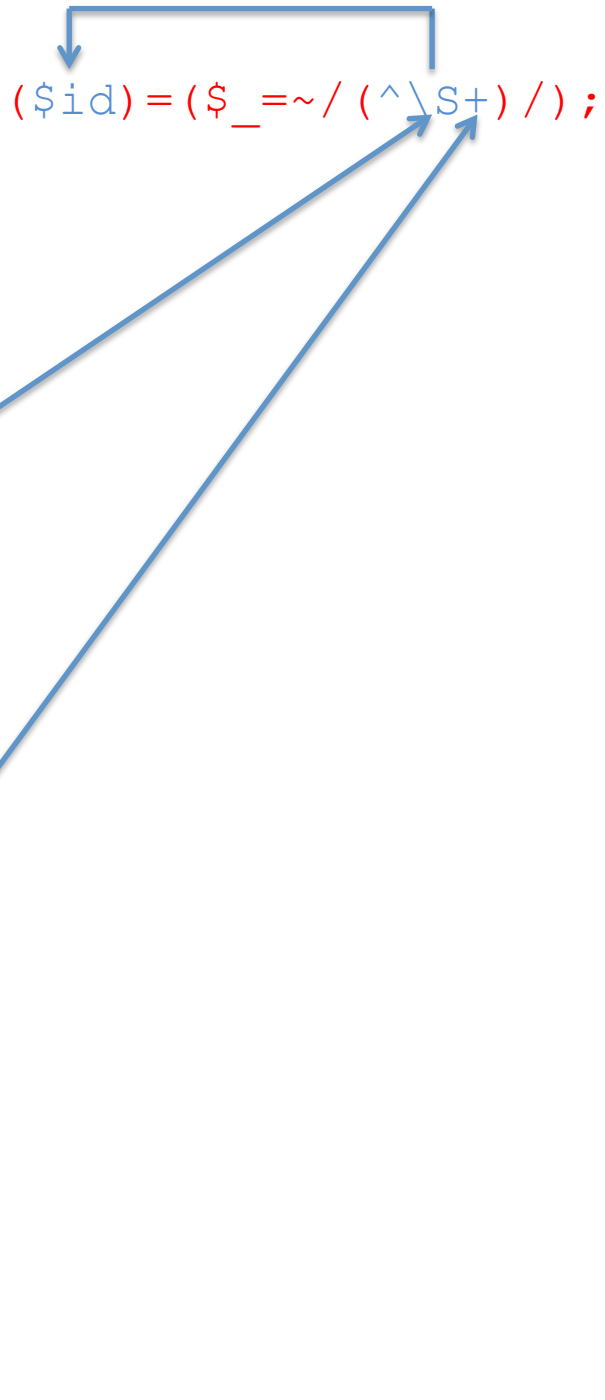
```
yyin@glu:~/work/class$ perl get-seq.pl test-query.fa.cowrument.out.my.head10.fa | less

gi|388476126|ref|YP_488309.1| homoserine kinase [Escherichia coli str. K-12 substr. W3110]      MVKVYAPASSANMSVGFDVLGAAVT
PVDGALLGDVVTVEAAETFSLNNLGRFADKLPSEPRENIVYQCWERFCQELGKQIPVAMTLEKNMPIGSGLGSSACSVVAALMAMNEHCGKPLNDTRLLALMGELEGRISGSIHYDNVAPC
FLGGMQLMIEENDIISQQVPGFDEWLWVLAYPGIKVSTAEARAILPAQYRRQDCIAHGRHLAGFIHACYSRQPELAAKLMKDVIAEPYRERLLPGFRQARQAVAEIGAVASGISGSGPTLF
ALCDKPETAQRVADWLGKNYLQNQEGFVHICRLDTAGARVLEN
NODE_3573325_length_256363_cov_8.500103_orf_03140 complement(202076..203146)    MKKIKVFAPASIANLGCGFDIMGMALDEVGDVLEMSLDEDS
SGISIVNETDVPLPEDIDQNVITPVIRKFFEMTGHSGRVDVRVLKKIYPGSGIGSSAASSAAAAFGINELFGAPLSEEDVVVCAMEGENLASGGYHADNAAPAVMGGIILIRGYEPLDVIK
LPVPGNLYCPVIHPHLMVSTKAARSILPKEIPMHTAITQWGNVGGLVAGLCTGNIELVGRAMRDAVAEPYRKGFIPGFDELRAKLLGAGALAMNISGSGPSVFALANRGDIAQRVGAIMER
HFAQQGILSETYVVKVXXXAAPQARGGRQALALRRQDGGRKVFRLPAGRREPAFLLCTGRXXXXSNKGARLIA*
NODE_3641687_length_21494_cov_1.063320_orf_137760 complement(19693..20634)    MKVSVRVPATVANIGPGFDCLGMALPIYNTITIEETVLPGT
GIEINVLANEDVTDELSLEHIPMDENSIIYKAVELLYNSIGQTPSELKITIHSEIPIAKGLGSSASVIVGGLIAANELLGKPADEAALLSIATEVEGHPDNITPAIIGGLTLSSAEEDGSI
VSRNLPWPEEWVLTVCVPEYELATEISRSVLPKEVPLTDAVYNAQRMAMFVQAIYTKDEELMKLALRDKLHQPYRMKLVPGFDKISENLKHEESVLGVVLSGAGPSILVVSLKTNLDKVKT
```

```
#!/usr/bin/perl -w

while(<>){
  chomp $_;  # get ride of newline character
  if($_=~/>/){  # =~ is used to match regexp
    $_=~s/>//;  # =~s is used to substitute
    ($id)=($_=~/(^\S+)/); # regexp, match a portion of text
                          # and capture the matched by ()
    print "\n",$id,"\t"; # insert a tabular space
  }
  else{
    print $_; # print the sequence line (no
        # newline character anymore)
  }
}
```

```
yyin@glu:~/work/class$ perl get-seq.pl test-query.fa.cowrument.out.m9.head10.fa | less

gi|388476126|ref|YP_488309.1|  MVKVYAPASSANMSVGFDVLGAAVTPVDGALLGDVVTVEAAETFSLNNLGRFADKLPSEPRENIVYQCWERFCQELGKQIPVAMTLEKN
MPIGSGLGSSACSVVAALMAMNEHCGKPLNDTRLLALMGELEGRISGSIHYDNVAPCFLGGMQLMIEENDIISQQVPGFDEWLWVLAYPGIKVSTAEARAILPAQYRRQDCIAHGRHLAGF
IHACYSRQPELAAKLMKDVIAEPYRERLLPGFRQARQAVAEIGAVASGISGSGPTLFALCDKPETAQRVADWLGKNYLQNQEGFVHICRLDTAGARVLEN
NODE_3573325_length_256363_cov_8.500103_orf_03140  MKKIKVFAPASIANLGCGFDIMGMALDEVGDVLEMSLDEDSSGISIVNETDVPLPEDIDQNVITP
VIRKFFEMTGHSGRVDVRVLKKIYPGSGIGSSAASSAAAAFGINELFGAPLSEEDVVVCAMEGENLASGGYHADNAAPAVMGGIILIRGYEPLDVIKLPVPGNLYCPVIHPHLMVSTKAAR
SILPKEIPMHTAITQWGNVGGLVAGLCTGNIELVGRAMRDAVAEPYRKGFIPGFDELRAKLLGAGALAMNISGSGPSVFALANRGDIAQRVGAIMERHFAQQGILSETYVVKVXXXAAPQA
RGGRQALALRRQDGGRKVFRLPAGRREPAFLLCTGRXXXXSNKGARLIA*
NODE_3641687_length_21494_cov_1.063320_orf_137760  MKVSVRVPATVANIGPGFDCLGMALPIYNTITIEETVLPGTGIEINVLANEDVTDELSLEHIPMD
ENSIIYKAVELLYNSIGQTPSELKITIHSEIPIAKGLGSSASVIVGGLIAANELLGKPADEAALLSIATEVEGHPDNITPAIIGGLTLSSAEEDGSIVSRNLPWPEEWVLTVCVPEYELAT
EISRSVLPKEVPLTDAVYNAQRMAMFVQAIYTKDEELMKLALRDKLHQPYRMKLVPGFDKISENLKHEESVLGVVLSGAGPSILVVSLKTNLDKVKTIIKETWDELSINAQMYTLPIDKTG
AVVIPE*
```

regexp metacharacters

Symbol	Meaning
.	any character
\w	alphanumeric and _
\W	any non-word character
\s	any whitespace
\S	any non-whitespace
\d	any digit character
\D	any non-digit character
\t	tab
\n	newline
*	match 0 or more times
+	match 1 or more times
?	match 0 or 1 times
{n}	match exactly n times
{n,m}	match n to m times
^	match from start
$	match to end



Use file handle to write to a file

```
#!/usr/bin/perl -w

open(OUT,">tmp");
while(<>){
  chomp $_;  # get ride of newline character
  if($_=~/>/){  # =~ is used to match regexp
    $_=~s/>//;  # =~s is used to substitute
    ($id)=($_=~/(^\S+)/); # regexp, match a portion of text
                          # and capture the matched by ()
    print OUT "\n",$id,"\t"; # insert a tabular space  }
  else{
    print OUT $_; # print the sequence line (no
                  # newline character anymore)
  }
}
close OUT;
```

```
perl get-seq.pl test-query.fa.cowrument.out.m9.head10.fa

less tmp
```

```
#!/usr/bin/perl -w

open(OUT,">tmp");
while(<>){
  chomp $_;  # get ride of newline character
  if($_=~/>/){  # =~ is used to match regexp
    $_=~s/>//;  # =~s is used to substitute
    ($id)=($_=~/(^\S+)/);
    print OUT "\n",$id,"\t"; # insert a tabular space
  }
  else{
    print OUT $_; # print the sequence line (no
newline character anymore)
  }
}
close OUT;

open(IN,"tmp");
while (<IN>){
  next if $_=~/^$/; # skip and go the next line
  chomp $_;
  @col=split(/\t/,$_);
  $seq_hash{$col[0]}=$col[1];
}
close IN;
print $seq_hash{'NODE_3573325_length_256363_cov_8.500103_orf_03140'}, "\n";
```

Convert fasta format to tabular format and then create hash

```
>gi|388476126|ref|
YP_488309.1| homoserine
kinase [Escherichia coli
str. K-12 substr. W3110]
MVKVYAPASSANMSVGFDVLGAAVTP
VDGALLGDVVTVEAAETFSLNNLGRF
ADKLPSEPRENIVYQCWE
RFCQELGKQIPVAMTLEKNMPIGSGL
GSSACSVVAALMAMNEHCGKPLNDTR
LLALMGELEGRISGSIHY
DNVAPCFLGGMQLMIEENDIISQQVP
GFDEWLWVLAYPGIKVSTAEARAILP
AQYRRQDCIAHGRHLAGF
IHACYSRQPELAAKLMKDVIAEPYRE
RLLPGFRQARQAVAEIGAVASGISGS
GPTLFALCDKPETAQRVA
DWLGKNYLQNQEGFVHICRLDTAGAR
VLEN
>NODE_3573325_length_25636
3_cov_8.500103_orf_03140
complement(202076..203146)
MKKIKVFAPASIANLGCGFDIMGMAL
DEVGDVLEMSLDEDSSGISIVNETDV
PLPEDIDQNVITPVIRKFFEMTGHSG
RVDV
```

```
yyin@glu:~/work/class$ perl get-seq2.pl test-query.fa.cowrument.out.m9.head10.fa
MKKIKVFAPASIANLGCGFDIMGMALDEVGDVLEMSLDEDSSGISIVNETDVPLPEDIDQNVITPVIRKFFEMTGHSGRVDVRVLKKIYPGSGIG
SSAASSAAAAFGINELFGAPLSEEDVVVCAMEGENLASGGYHADNAAPAVMGGIILIRGYEPLDVIKLPVPGNLYCPVIHPHLMVSTKAARSILP
KEIPMHTAITQWGNVGGLVAGLCTGNIELVGRAMRDAVAEPYRKGFIPGFDELRAKLLGAGALAMNISGSGPSVFALANRGDIAQRVGAIME
RHFAQQGILSETYVVKVXXXAAPQARGGRQALALRRQDGGRKVFRLPAGRREPAFLLCTGRXXXXSNKGARLIA*
```

```
#!/usr/bin/perl -w

open(DB,$ARGV[0]);
open(OUT,">tmp");

while(<DB>){
  chomp $_;  # get ride of newline character
  if($_=~/>/){  # =~ is used to match regexp
    $_=~s/>//;  # =~s is used to substitute
    ($id)=($_=~/(^\S+)/);
    print OUT "\n",$id,"\t"; # insert a tabular space
  }
  else{
    print OUT $_; # print the sequence line
  }
}
close OUT; close DB;

open(IN,"tmp");
while (<IN>){
  next if $_=~/^$/;
  chomp $_;
  @col=split(/\t/,$_);
  $seq_hash{$col[0]}=$col[1];
}
close IN;
open(ID,$ARGV[1]);
while(<ID>){
  chomp $_;
  print ">$_\n",$seq_hash{$_},"\n";
}
close ID; system("rm tmp");
```

There are 2,547,270 fasta sequences

Loaded into memory, id as KEY, seq as VALUE

There are 104 seq ids, one id per line

Now the id is called to return the seq

Step 1: prepare the tabular format file (id + seq)
Step 2: load the tabular format file into memory as a hash
Step 3: read in the id file and extract seqs

```
perl get-seq3.pl metagenemark_predictions.faa test-query.fa.cowrument.out.m9.hitid
| less
```

Perl one-liner

You don't write codes into a file and then issue "`perl file.pl`" on the command line;
You write the codes directly on the command line, like you are typing regular Linux commands

```
#!/usr/bin/perl

open (IN,$ARGV[0]);

@a=<IN>;

foreach(@a){
   @col=split(/\t/,$_);
   print $col[1],"\tmutation\n";
}
```

=

```
#!/usr/bin/perl

while (<>){
   @col=split(/\t/,$_);
   print $col[1],"\tmutation\n";
}
```

```
perl -e 'while(<>){@col=split(/\t/,$_);print
$col[1],"\tmutation\n";}' cosmicRaw.txt.head10.6col
```

=

```
cat cosmicRaw.txt.head10.6col |
cut -f2 | awk '{print
$1,"mutation"}' | sed 's/ /\t/'
```