References

Abbot, J. C. et al. (2005) *Bioinformatics* 21(18) 3665-3666.
WebACT – an online companion for the Artemis Comparison Tool

Berriman, M., and K. Rutherford (2003) *Brief Bioinform* 4 (2) 124-132.
Viewing and annotating sequence data with Artemis.

Carver T. J. et al. (2010) *Bioinformatics* (doi:10.1093/bioinformatics/btq010).
BamView: Viewing mapped read alignment data in the context of the reference sequence.

Carver T.J. et al. (2012) *Bioinformatics* 28 (4): 464-9.
Artemis: an integrated platform for visualization and analysis of high-throughput sequence-based experimental data.

Carver T.J. et al. (2012) *Brief Bioinform* (doi: 10.1093/bib/bbr073).
BamView: visualizing and interpretation of next-generation sequencing read alignments.

Carver T. J. et al. (2005) *Bioinformatics* 21: 3422-3.
ACT: the Artemis Comparison Tool.

Fidock, D.A. et al. (2000) *Molecular Cell* 6: 861-871.
Mutations in the *P. falciparum* digestive vacuole transmembrane protein PfCRT and evidence for their role in chloroquine resistance.

Logan-Klumpler, F.J. et al (2012) *Nucleic Acid Res* 40; D98-108.
GeneDB--an annotation database for pathogens.

Rutherford et al. (2000) *Bioinformatics* 16 (10) 944-945.
Artemis: sequence visualization and annotation.

Sanchez, C. and Lanzer, M. (2000) *Current Opinion on Infectious Diseases* 13**:** 653-658.
Changing ideas on chloroquine in *Plasmodium falciparum*.

Appendices

Appendix I: Course Virtual Machine (VM) Quick Start Guide

Using a VM enables us to encapsulate the course data and software in such a way that you can still make use of them when you return to your own laboratory.

To use the VM on the USB stick provided, you will first need to download VirtualBox (http://www.virtualbox.org/). This software is required to run the VM on your machine, it is free and available for windows, MacOSX and linux,

For a detailed description of VirtualBox and the installation see the on-line manual (http://www.virtualbox.org/manual/).

Download and Install VirtualBox

Download VirtualBox for the type of workstation you are using (e.g. windows) from http://www.virtualbox.org/wiki/Downloads.
Double click on the executable file (windows). The installation welcome dialog opens and allows you to choose where to install VirtualBox to and which components to install. Depending on your Windows configuration, you may see warnings about "unsigned drivers" or similar. Please select "Continue" on these warnings; otherwise VirtualBox might not function correctly after installation.
Launch the VirtualBox software from the desktop shortcut or from the program menu.

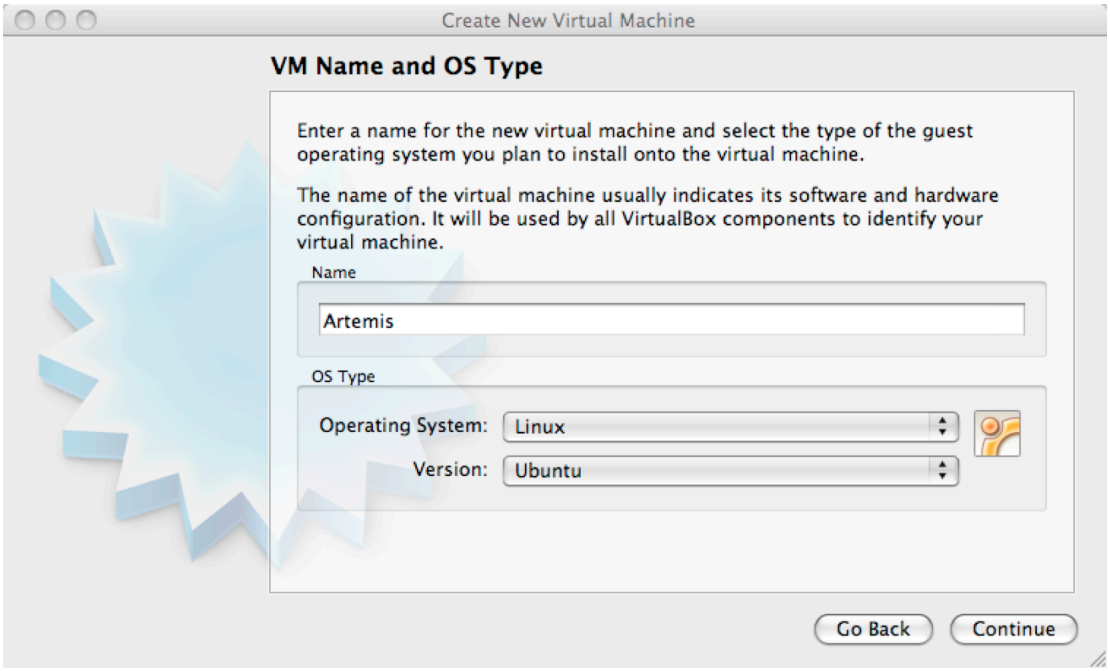
Setting up the VM

Download the image from
ftp://ftp.sanger.ac.uk/pub/project/pathogens/tdo/London/VM.zip

Save it and unzip it.

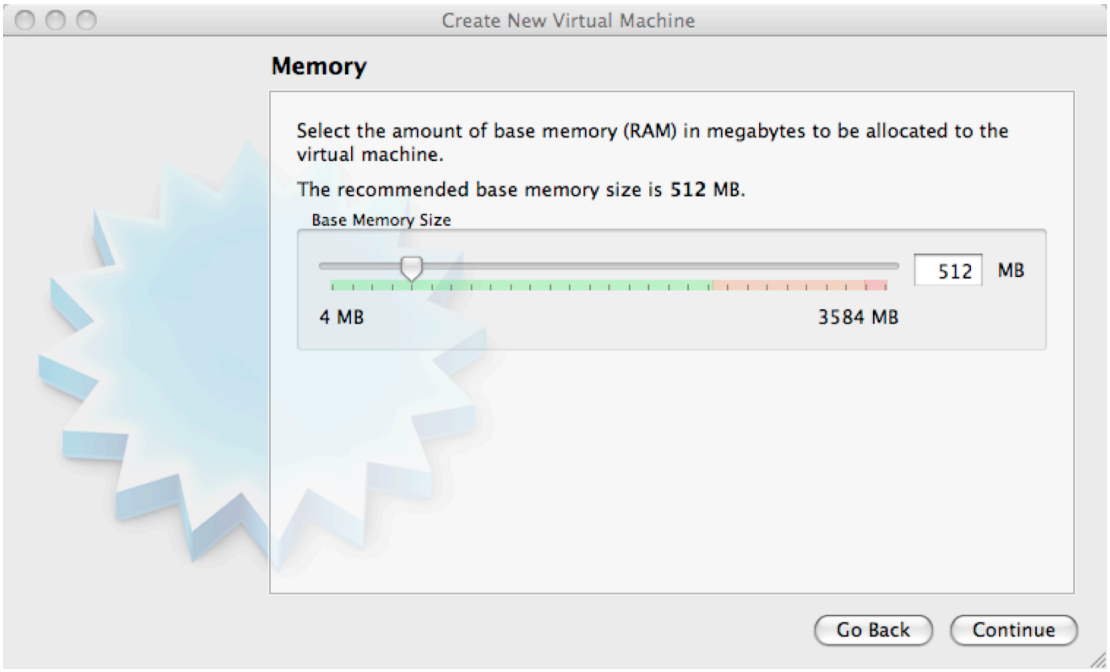
Create a new virtual machine by selecting ‘New’ from the options at the top.

Then fill the boxes in as shown:
In the first window enter:
Name: **Artemis**
Operating System: **Linux**
Version: **Ubuntu**



- Click ‘Continue’

- In the next window keep the memory default setting (512 MB)

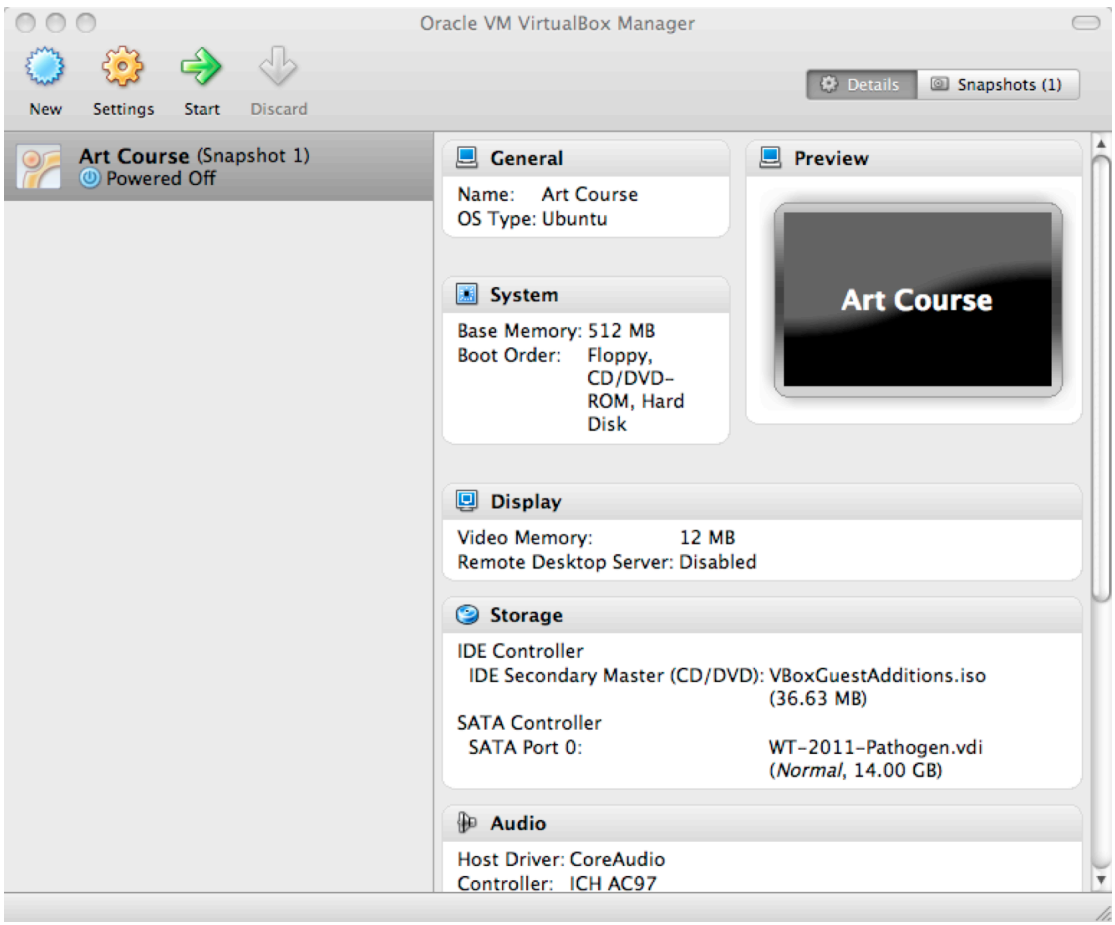


Click ‘Continue’.

In the next window select ‘Use existing hard disk’ and from the folder icon on the right hand side navigate to the memory USB stick and select the VDI file located on the memory stick.

- Click ‘Continue’

- There will now be an ‘Artemis’ (powered off) button in the left hand side of VirtualBox.



Double click on this new Artemis course power button to start the VM. The VM is designed to look for a shared folder between the operating system you are running it on and the VM operating system. So when the VM is first started after a few seconds you will see the screen below and you should type ‘s’ to skip this part.

It will then log you into the Ubuntu desktop.

Setting up a Shared Folder

This allows you to share a folder between the VM and your workstation. This means you can put files that you want to share between the operating systems in this folder.

Create a directory to share called ‘VMshare’ on your machine. With the VM shutdown select the ‘Artemis’ button in VirtualBox and click ‘Settings’ in the top menu bar. Go to ‘Shared Folders’ and select the ‘+’ button on the right. In the ‘Folder Path’ select ‘Other’ and navigate to and select the ‘VMshare’ folder that you have created. Then click on ‘OK’. When the ‘Artemis’ VM is next started it will show the contents of this folder in the /home/wt/host directory in Ubuntu.

The root passoword is **wt**.

Appendix II: Blast commands

formatdb -p F -i <database.fasta>
blastall -p tblastx -m 8 -e 1e-20 -m 8 -d <database.fasta> -i <query.fasta> -o comp.<name>.blast

Appendix II: Artemis minimum hardware and software requirements.

Artemis and ACT will, in general, work well on any standard modern machine and with most common operating systems. It is currently used on many different varieties of UNIX and Linux systems as well as Apple Macintosh and Microsoft Windows systems.
Note that the ability to run external programs (such as BLAST and FASTA) from within Artemis and ACT is available only on UNIX and Linux systems.
Minimum memory requirements for people working on whole genomes are approximately 128 megabytes for Artemis and 128 megabytes per genome for ACT. Analysis of cosmid sized sequences can comfortably be achieved with less memory.

Appendix III: ACT comparison files

ACT supports three different comparison file formats:

1) BLAST version 2.2.2 output: The blastall command must be run with the -m 8 flag which generates one line of information per HSP.
2) MegaBLAST output: ACT can also read the output of MegaBLAST, which is part of the NCBI blast distribution.
3) MSPcrunch output: MSPcrunch is program for UNIX and GNU/Linux systems which can post-process BLAST version 1 output into an easier to read format. ACT can only read MSPcrunch output with the -d flag.

Here is an example of an ACT readable comparison file generated by MSPcrunch -d.

```
1399 97.00 940 2539 sequence1.dna 1 1596 AF140550.seq
1033 93.00 9041 10501 sequence1.dna 9420 10880 AF140550.seq
828 95.00 6823 7890 sequence1.dna 7211 8276 AF140550.seq
773 94.00 2837 3841 sequence1.dna 2338 3342 AF140550.seq
```

The columns have the following meanings (in order): score, percent identity, match start in the query sequence, match end in the query sequence, query sequence name, subject sequence start, subject sequence end, subject sequence name.

The columns should be separated by single spaces.

Appendix IV: Feature Keys and Qualifiers – a brief explanation of what they are and a sample of the one's we use.

1 – Feature Keys: They describe features with DNA coordinates and once marked, they all appear in the Artemis main window. The ones we use are:

CDS: Marks the extent of the coding sequence.
RBS: Ribosomal binding site
misc_feature: Miscellaneous feature in the DNA
rRNA: Ribosomal RNA
repeat_region
repeat_unit
stem_loop
tRNA: Transfer RNA

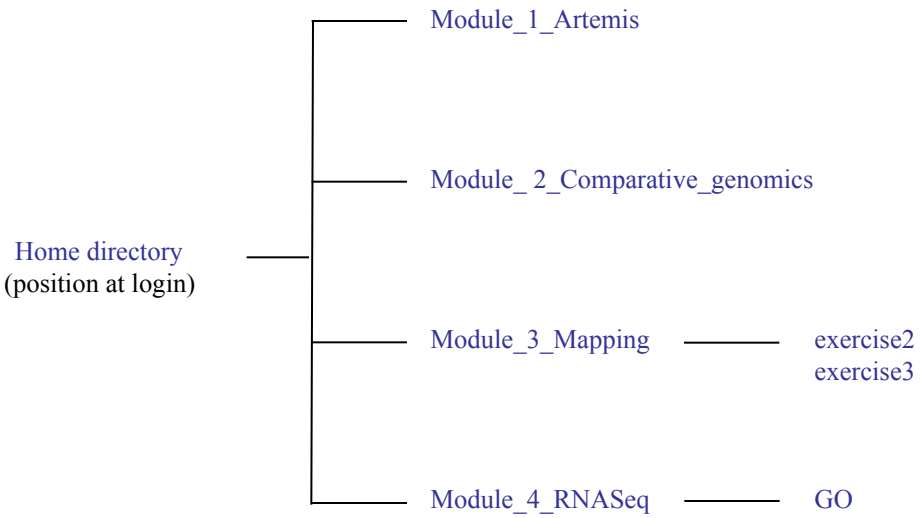
2 – Qualifiers: They describe features with protein coordinates. Once marked they appear in the lower part of the Artemis window. They describe the gene whose coordinates appear in the 'location' part of the editing window. The ones we commonly use for annotation at the Sanger Institute are:

colour: Also used in-house in order to differentiate between different types of genes and other features.
gene: Descriptive gene a name, eg. ilvE, argA, VAR etc.
label: Allows you to label a gene/feature in the main view panel.
note: This qualifier allows for the inclusion of free text. This could be a description of the evidence supporting the functional prediction or other notable features/information which cannot be described using other qualifiers.
product: The assigned possible function for the protein goes here.
pseudo: Matches in different frames to consecutive segments of the same protein in the databases can be linked or joined as one and edited in one window. They are marked as pseudogenes. They are normally not functional and are considered to have been mutated.
locus_tag : Systematic gene number, eg SAS1670, Sty2412 etc.

The list of keys and qualifiers accepted by EMBL in sequence/annotation submission files are listed at the following web page:
http://www.ebi.ac.uk/ena/WebFeat/

Appendix V: Schematic of workshop files and directories

Key:
Directories and subdirectories



Appendix VI: Useful Web addresses

Major Public Sequence Repositories

DNA Data Bank of Japan (DDBJ)	http://www.ddbj.nig.ac.jp
EMBL Nucleotide Sequence Database	http://www.ebi.ac.uk/ena/
Genomes at the EBI	http://www.ebi.ac.uk/genomes
GenBank	http://www.ncbi.nlm.nih.gov/genbank

Genome Databases Resources

GeneDB	http://www.genedb.org
Eupathdb	http://eupathdb.org/

Protein Motif Databases

Prosite	http://prosite.expasy.org/
Pfam	http://pfam.xfam.org/
InterPro	http://www.ebi.ac.uk/interpro/
PRINTS	http://www.bioinf.manchester.ac.uk/dbbrowser/PRINTS/
SMART	http://smart.embl-heidelberg.de

Protein feature prediction tools

TMHMM Transmembrane helices prediction	http://www.cbs.dtu.dk/services/TMHMM/
SignalP Prediction Server	http://www.cbs.dtu.dk/services/SignalP/
PSORT protein prediction	http://psort.hgc.jp/form.html

Metabolic Pathways and Cellular Regulation

ENZYME	http://www.expasy.ch/enzyme/
Kyoto Encyclopedia of Genes and Genomes (KEGG)	http://www.genome.jp/kegg/
MetaCyc	http://metacyc.org/

Miscellaneous sites

NCBI BLAST website	http://http://blast.ncbi.nlm.nih.gov/Blast.cgi
EBI FASTA website	http://www.ebi.ac.uk/Tools/sss/fasta/
tRNAscan-SE Search Server	http://selab.janelia.org/tRNAscan-SE/
Rfam	http://rfam.xfam.org/
Codon usage database	http://www.kazusa.or.jp/codon/
GO Gene Ontology Consortium	http://geneontology.org/
Artemis homepage	http://www.sanger.ac.uk/resources/software/artemis/
ACT homepage	http://www.sanger.ac.uk/resources/software/act/
WebACT	http://www.webact.org/WebACT/home
Double ACT	http://www.hpa-bioinfotools.org.uk/pise/double_act.html
Glimmer	http://ccb.jhu.edu/software/glimmer/index.shtml
EasyGene	http://www.cbs.dtu.dk/services/EasyGene/
String	http://string.embl.de
EMBOSS	http://emboss.sourceforge.net/

Appendix VII: List of colour codes

2 (red) - gene product is based on experimental evidence
6 (dark pink) - unlikely hypothetical protein
7 (yellow) - gene product is based on orthology
8 (light green) - hypothetical protein, unknown function
10 (orange) - conserved hypothetical
13 (light grey) - pseudogenes

Appendix VIII: List of degenerate nucleotide value/IUB Base Codes.

R = A or G

S = G or C

B = C, G or T

Y = C or T

W = A or T

D = A, G or T

K = G or T

N = A, C, G or T

H = A, C or T

M = A or C

V = A, C or G

Appendix VIIII Splice site information

Gene	No.	Exon	Intron	Exon	Size (bp)
41-3	1	GAA	**GTA**CACA..CCTTCTTTTTCCATATT	**TAG**|CAA	152
	2	AAT	**GTT**AAAA...TTTTTTTTTTTAAACT	**TAG**|CCG	208
	3	GAG	**GTA**AGAA...ATTCATTATATATTTA	**TAG**|GGA	86
	4	TCG	**GTA**TGGA...TTTTGAAATACTTCCT	**CAG**|TTA	152
	5	ACT	**GTA**ATAT..TTTTTTTTTTTATTTCC	**TAG**|ATG	112
	6	CAG	**GTA**AATA..ATAATGACATTTTGATA	**CAG**|ATT	120
	7	AAT	**GTA**CATT..TTATTTTTATTTATTTA	**TAG**|AAA	81
	8	TAG	**GTA**TTTG..ATATTTTTTACTTATGA	**TAG**|TTA	96
RhopH3	1	AGG	**GTA**ATAT..TTTATTTTATTTTTTTT	**TTA**|TTT	150
	2	GGA	**GTA**AGAG..TTTTTATTATTTTATTG	**TAG**|TCC	442
	3	GGA	**GTA**AGAG..TTTTTATTATTTTATTG	**TAG**|TCC	199
	4	CAG	**GTA**YGCT..TTTAATTTTTTTTTCCT	**TCA**|TCA	160
	5	AAA	**GTA**AGAA..TATTTTTTTACAATTTT	**TAG**|TTC	206
	6	AAG	**GTA**AAAG..TTTTTTTTTTTTTGTTT	**CAG**|TTT	142
RNA pol III	1	CAG	**GTA**CATA..TTTTTTTTTTTTTTTTT	**TAG**|GTG	158
	2	CAA	**GTA**ATTA..TATATTTTATTTTTTCT	**TAG**|GTT	113
	3	TAC	**GTT**AGTT..TTTTTTTTTTTTTTTTT	**TAG**|TGG	169
	4	ATT	**GTA**AGTT..TATTTTTTTTTTTTTTT	**TAG**|TGA	112
SERA	1	TGT	**GTA**AGAA..TTGTCATTATTTTTTTT	**TAG**|GTG	158
	2	AAA	**GTA**TAAA..TTTATTTATTTTTTTTT	**TAG**|ATA	175
	3	CAG	**GTA**AATA..TTTTAATTTTTTTGTTT	**TAG**|AAA	129
SERP H	1	CTG	**GTT**TGTC..CATATATTTCTTTATTT	**TAG**|ATA	345
	2	AGA	**GTA**AAAA..TTTCTTATATTTTCTTT	**TAG**|GTG	92
	3	CTG	**GTT**TGTC..CATATATTTCTTTATTT	**TAG**|ATA	116
Ag15	1	ATG	**GTA**AGAG..TATTTTTGATACCTTTA	**TAG**|AGT	214
	2	AAA	**GTA**ATTA..CAATCATATTAACACAA	**AAG**|ATG	280
PfGPx	1	GAG	**GTA**TACA..TTATTATTCCCTTGCTT	**TAG**|ATC	208
	2	TCG	**GTT**AGTA..TATTTATCATTTTTTTC	**CAG**|ATG	168
Calmodulin	1	GAA	**GTA**AATC..TTTTTTATTTTTCTCAT	**TAG**|CTA	480
PfPK1	1	TAG	**GTG**TGTT..TCATTACATTTTTACCT	**TAG**|GAT	101
MESA	1	TTA	**GTA**AGTT..CGTAATATATTTTTTTTT	**TAG**|GAT	122
Aldolase	1	ATG	**GTA**AGAA..TATTTTTATATTTTTTT	**TAG**|GCT	452
KAHRP	1	AAC	**GTA**AGTT..TTATTTTTTTTTTCATA	**TAG**|TGC	430
GBPH2	1	TTG	**GTA**TGCC..TTTGTATTATTTAATTT	**TAG**|AAT	157
GBP	1	TTG	**GTA**TG....TGTGTATTGTTTATTTT	**TAG**|AAT	179
FIRA	1	TGT	**GTA**AGGA..TTTTTATATTTTTTCTT	**TAG**|CGA	175
GARP	1	AAG	**GTA**ACAA..TATATGTATTTTTTTTT	**TAG**|TGC	214

Donor motif **Acceptor motif**

The splice acceptor and donor sequences for several *P. falciparum* genes: adapted from Coppel and Black(1998). In "Malaria:Parasite Biology, Pathogenesis and Protection", I.W. Sherman (ed.); ASM Press; Washington DC; pp185-202

BASIC UNIX

Introduction

Unix is the standard operating system on most large computer systems in scientific research, in the same way that Microsoft Windows is the dominant operating system on desktop PCs.

Unix and MS Windows both perform the important job of managing the computer's hardware (screen, keyboard, mouse, hard disks, network connections, etc) on your behalf. They also provide you with tools to manage your files and to run application software.

They both offer a graphical user interface. On Unix systems, this is called the X Window System, or just X.

Unix is a powerful, robust and stable operating system which allows dozens of people to run programs on the same computer at the same time. This is why it is the preferred operating system for large-scale scientific computing. It runs on all kinds of machines, from desktop PCs to supercomputers.

Aims

The aim of this module is to introduce UNIX and cover some of the basics that will allow you to run some of the programs used in this workshop. Several of the programs that you are going to use during the workshop, plus many others that are useful for bioinformatics analyses, are run in UNIX. This module is only designed to provide a very brief introduction to some of the features and useful commands of UNIX.

During this module we will also obtain a genome sequence that will be used in the next module, and examine the basic structure of an EMBL entry.

Why use UNIX?

• UNIX is a well established, very widespread operating system.

• Command line driven, with a huge number of often terse, but powerful commands

• In contrast to Windows, it is designed to allow many users to run their programs simultaneously on the same computer

• Designed to work in computer networks - for example, most of the Internet is UNIX based

• It is used on many of the powerful computers at bioinformatics centres and also on many workstations

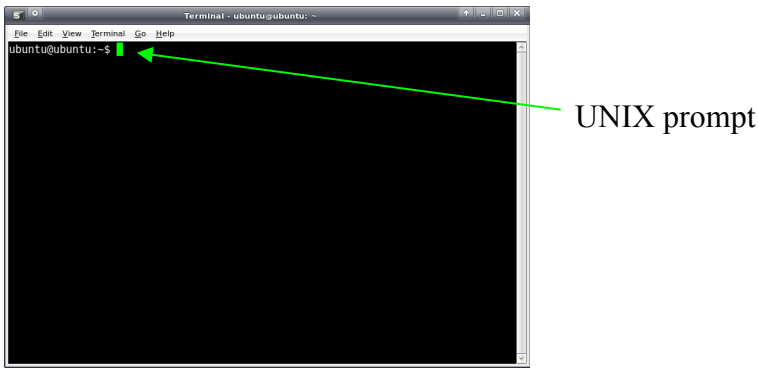
• UNIX is not a monolithic entity. There are numerous different UNIX operating systems. Some of them are freely distributed such as Linux which was originally created to provide a free UNIX on personal PCs. This operating system is now so popular that it has been ported to many different system architectures.

Getting started

In this workshop, we will be using desktop PCs which run Linux, a version of UNIX which was specially designed for PCs.

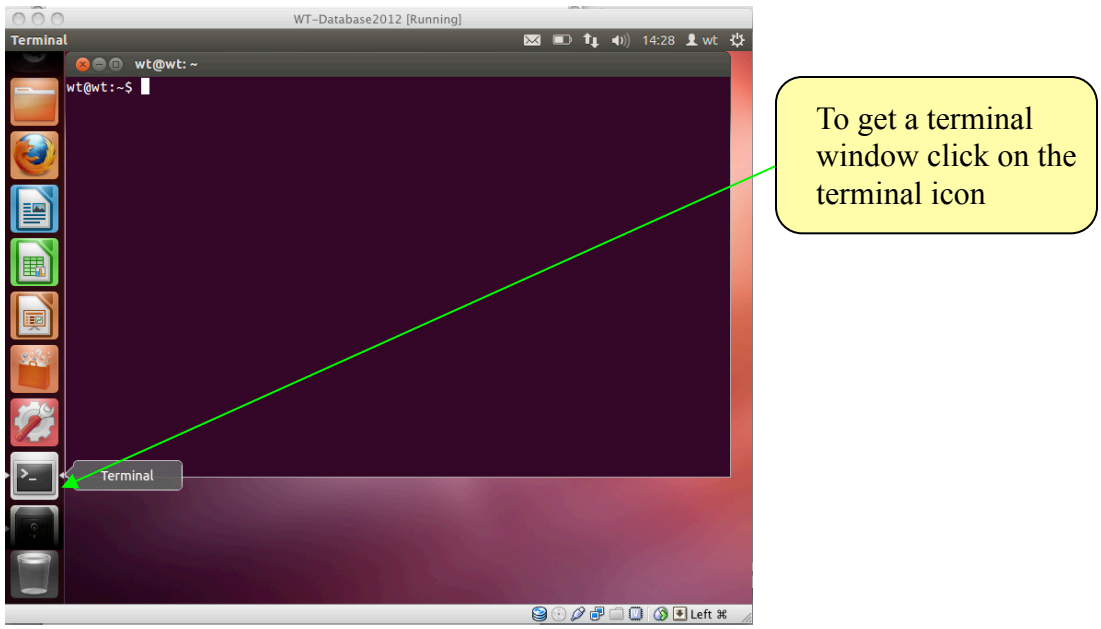
We will use a terminal window to type in our UNIX command line. This is similar to the "Command Prompt" window on MS Windows systems, which allows the user to type DOS commands to manage files.

You should see a window labelled "Terminal" which will be empty except for a ’$' character at the top left. The '$' character is the UNIX prompt, similar to "C:\" in DOS. Note: the prompt will often be different on different UNIX computers, for example it may be displayed as a ‘%’ character.



You can type commands directly into the terminal at the ‘%' prompt.

A list of useful commands can be found on the next page.
Many of them are two- or three-letter abbreviations. The earliest UNIX systems (*circa* 1970) only had slow Teletype terminals, so it was faster to type 'rm' to remove a file than 'delete' or 'erase'. This terseness is a feature of UNIX which still survives.



The command line

All UNIX programs may be run by typing commands at the UNIX prompt **$**. The command line tells the computer what to do.
You may subtly alter these commands by specifying certain options when typing in the command line.

Command line Arguments
Typing any of the commands listed above at the UNIX prompt with the appropriate variables such as files names or directories will result in the tasks being performed on pressing the enter key.

Additional arguments or flags can be added to the commands to affect the way the command works. For example:

The **cal** command prints a calendar for a month or a year

If you type in just **cal** with no month or year, you get the calendar for the current month

If you type **cal** and a year you get the calendar for that year

$ **cal 2000** [enter]

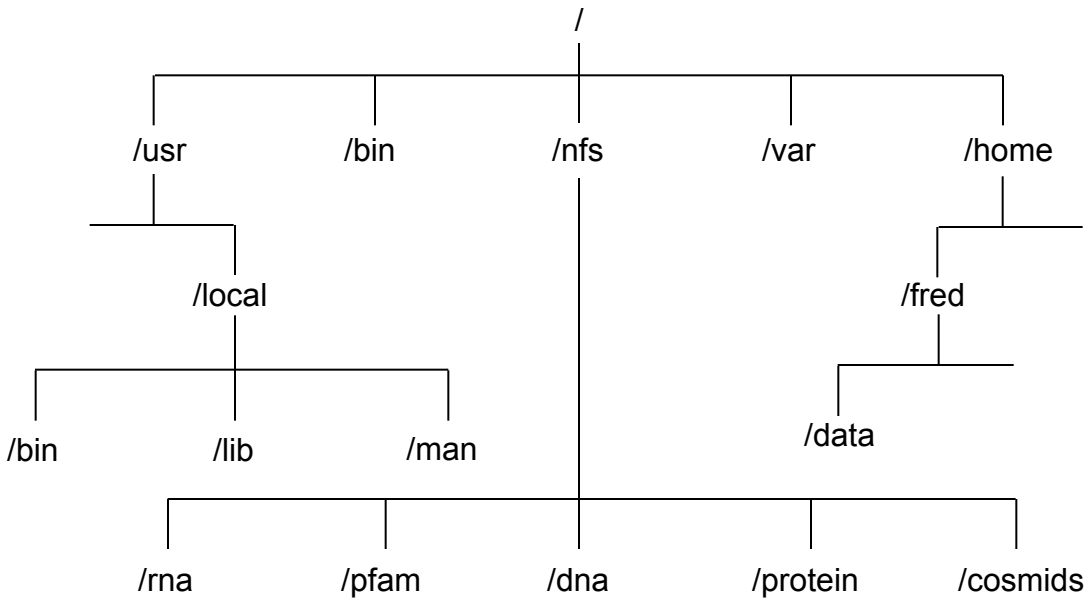
Additional arguments for the commands are not covered here, but if you want to find out what arguments are available, or want to find out more about a UNIX command, type **man** followed by the UNIX command

$ **man cal** [enter]

Files and Directories

Directories are the UNIX equivalent of folders on a PC or Mac. They are organised in a hierarchy, so directories can have sub-directories and so on. Directories are very useful for organising your work and keeping your account tidy - for example, if you have more than one project, you can organise the files for each one into different directories to keep them separate. You can think of directories as rooms in a house. You can only be in one room (directory) at a time. When you are in a room you can see everything in that room easily. To see things in other rooms, you have to go to the appropriate door and crane your head around. UNIX works in a similar manner, moving from directory to directory to access files. The location or directory that you are in is referred to as the current working directory.

Directory structure example



Therefore if there is a file called genome.seq in the **dna** directory its location or full pathname can be expressed as /nfs/dna/genome.seq.

For the actual directory structure you will be using during the workshop, see Appendix IV.

General Points

UNIX is pretty straightforward, but there are some general points to remember that will make your life easier:

UNIX is case sensitive - typing "ls" is not the same as typing "LS".

You need to put a space between a command and its argument - for example, "more myfile" will show you the contents of the file called myfile; "moremyfile" will just give you an error!

UNIX is not psychic! If you mis-spell the name of a command or the name of a file, it will not understand you.

Many of the commands are only a few letters long; this can be confusing until you start to think logically about why those letters were chosen - ls for list, rm for remove and so on.
Often when you have problems with UNIX, it is due to a spelling mistake, or perhaps you have omitted a space.

If you want to know more about UNIX and its commands there are plenty of resources available that provide a more comprehensive guide:-
(e.g. http://unixhelp.ed.ac.uk or http://unix.t-a-y-l-o-r.com/).

In what follows, we shall use the following typographical conventions:

Characters written in **`bold typewriter font`** are commands to be typed into the computer as they stand.

Characters written in *`italic typewriter font`* indicate non-specific file or directory names.

Words inserted within square brackets `[Ctrl]` indicate keys to be pressed.

So, for example,

$ **`ls`** *`anydirectory`* `[Enter]`

means "at the UNIX prompt $, type ls followed by the name of some directory, then press the key marked Enter"

Don't forget to press the [Enter] key: commands are not sent to the computer until this is done.

Some useful UNIX commands

Command	What it does
ls	Lists the contents of the current directory
mkdir	Makes a new directory
mv	Moves or renames a file
cp	Copies a file
rm	Removes a file
cat	Concatenates files
more	Displays the contents of a file one page at a time
head	Displays the first ten lines of a file
tail	Displays the last ten lines of a file
cd	Changes current working directory
pwd	Prints working directory
find	Finds files matching an expression
grep	Searches a file for patterns
wc	Counts the lines, words, characters, and bytes in a file
kill	Stops a process
jobs	Lists the processes that are running

Exercise

The following exercise introduces a few useful UNIX commands and provides examples of how they can be used.

Many people panic when they are confronted with an UNIX prompt! Don't! The exercise is designed to be step-by-step, so all the commands you need are provided in the text. If you get lost ask a demonstrator. If you are a person skilled at UNIX, be patient it is only a short exercise.

Finding where you are and what you've got

pwd Print the working directory

As seen previously directories are arranged in a hierarchical structure. To determine where you are in the hierarchy you can use the **pwd** command to display the name of the current working directory. The current working directory may be thought of as the directory you are in, i.e. your current position in the file-system tree

To find out where you are type

```
pwd [enter]
```

You will see that you are in your home directory. We need to move into the UNIX directory

UNIX is case sensitive, **PWD** is not the same as **pwd**

cd Change current working directory

The **cd** command will change the current working directory to another, in other words allow you to move up or down in the directory hierarchy. First of all we are going to move into the UNIX directory below. To do this type:

```
cd UNIX [enter]
```

Now use the **pwd** command to check your location in the directory hierarchy.

ls List the contents of a directory

To find out what are the contents of the current directory type

```
ls [enter]
```

The ls command lists the contents of your current directory, this includes files and directories
You should see that there are 4 files called:
AL513382.embl, MAL13P1.dna, MAL13P1.tab, Malaria.fasta

Changing and moving what you've got

cp Copy a file

cp file1 file2 is the the command which makes a copy of file1 in the current working directory and calls it file2

What you are going to do is make a copy of AL513382.embl. This file contains the genome of *Salmonella typhi* strain CT18 in EMBL format. The new file will be called S_typhi.embl

```
cp AL513382.embl  S_typhi.embl [enter]
```

If you use the **ls** command to check the contents of the current directory you will see that there are now two files, AL513382.embl and S_typhi.embl.

rm Delete a file

This command removes a file permanently so take care!

You are now going to remove the old version of *S. typhi* genome file, AL513382.embl

```
rm AL513382.embl [enter]
```

The file will be removed. Use the **ls** command to check the contents of the current directory to see that AL513382.embl has been removed.

UNIX as a general rule does exactly what you ask, and does not ask for confirmation. Unfortunately there is no "recycle bin" on the command line to recover the file from, so you have to be careful.

cd Change current working directory

As before the **cd** command will change the current working directory to another, in other words allow you to move up or down in the directory hierarchy. First of all we are going to move into the directory above, type:

```
cd .. [enter]
```

Now use the **pwd** command to check your location in the directory hierarchy.

Next, we are going to move into the Module_1_Artemis directory.

To change to the Module_1_Artemis directory type:

cd *Module_1_Artemis* [enter]

use the **ls** command to check the contents of the directory

mv Move a file

To move a file from one place to another use the **mv** command. This moves the file rather than copies it, therefore you end up with only one file rather than two.

When using the command the path or pathname is used to tell UNIX where to find the file. You refer to files in other directories by using the list of hierarchical names separated by slashes. For example, the file *bases* in the directory *genome* has the path **genome/bases**

If no path is specified UNIX assumes that the file is in the current working directory.

What you are going to do is move the file S_typhi.embl from the UNIX directory, to the current working directory

mv *../UNIX/S_typhi.embl* **.** [enter]

Use the **ls** command to check the contents of the current directory to see that S_typhi.embl has been moved.

../UNIX/S_typhi.embl specifies that S_typhi.embl is in the UNIX directory. If the file was in the directory above, the path would change to: ../S_typhi.embl

. specifies the current working directory
.. specifies the directory above the current working direcory

The command can also be used to rename a file in the current working directory. Previously we used the **cp** command, but **mv** provides an alternative without the need to delete the original file. Therefore we could have used:

mv *AL513382.embl S_typhi.embl* [enter]

Viewing what you've got

more Display file contents

This command displays the contents of a specified file one screen at a time.

You are now going to look at the contents of S_typhi.embl.

more *S_typhi.embl* [enter]

The contents of S_typhi.embl will be displayed one screen at a time, to view the next screen press the **space bar**. The percentage of the file that has been viewed so far will be displayed at the bottom of the screen. As S_typhi.embl is a large file this will take a while, therefore you may want to escape or exit from his command. To do this press the **control** and **c** keys simultaneously, this kills the **more** command, and returns you to the UNIX prompt. **more** can also scroll backwards if you hit the **b** key. Another useful feature is the **slash key**, /, to search for an expression in the file.

head Display the first ten lines of a file
tail Display the last ten lines of a file

Sometimes you may just want to view the text at the beginning or the end of a file, without having to display all of the file. The head and tail commands can be used to do this.

You are now going to look at the beginning of S_typhi.embl.

head *S_typhi.embl* [enter]

To look at the end of S_typhi.embl type:

tail *S_typhi.embl* [enter]

The amount of the file that is displayed can be increased by adding extra arguments. To increase the number of lines viewed from 10 to 100 add the –100 argument to the command. For example to view the last 100 lines of S_typhi.embl type:

tail -100 *S_typhi.embl* [enter]

Do this for both head and tail commands. What type of information is at the beginning and end of the EMBL format file?

cat Join files together

Having looked at the beginning and end of the S_typhi.embl file you should notice that in EMBL format files the annotation comes first, then the DNA sequence at the end.

If you had two separate files containing the annotation and the DNA sequence, both in EMBL format, it is possible to concatenate or join the two together to make a single file like the S_typhi.embl file you have just looked at. The UNIX command **cat** can be used to join two or more files into a single file. The order in which the files are joined is determined by the order in which they appear in the command line.

For example, we have two separate files, MAL13P1.dna and MAL13P1.tab, that contain the DNA and annotation, respectively, from the *P. falciparum* genome.

By typing the command line:

```
cat MAL13P1.tab MAL13P1.dna > MAL13P1.embl [enter]
```

MAL13P1.tab and MAL13P1.dna will be joined together and written to a file called MAL13P1.embl

The **>** symbol in the command line directs the output of the **cat** program to the designated file MAL13P1.embl

wc Counts the lines, words or characters

By typing the command line:

```
ls | wc –l [enter]
```

The above command uses **wc** to count the number of files that are listed by **ls**.

The | symbol in the command line connects the two commands into a single operation for simplicity. You can connect as many commands as you want:

$ **ls** | **grep** "**.embl**" | **wc** -l

grep Searches a file for patterns

grep is a powerful tool to search for patterns in a file.

In the examples below, we are going to use the file called Malaria.fasta that contains the set of P. falciparum chromosomes in FASTA format. A FASTA file has the following format:

>Sequence Header
CTAAACCTAAACCTAAACCCTGAACCCTAA…
…

Therefore if we want to get the sequence headers, we can extract the lines that match the > symbol:

```
grep '>' Malaria.fasta [enter]
```

By typing the command line:

```
grep -c '>' Malaria.fasta [enter]
```

The > symbol is placed in quotes as this stops the shell from interpreting the > as an instruction for where to put the output.
The -c option prints only a count of matching lines. Therefore in this example we will display the number of sequence entries that this file contains.

find Finds files matching an expression

The **find** command is similar to **ls** but in many ways it is more powerful. It can be used to recursively search the directory tree for a specified path name, seeking files that match a given Boolean expression (a test which returns true or false)

find . -name "*.embl"

This command will return the files which name has the .embl suffix.

find . -type d

This command will return all the subdirectories contained in the current directory.

These is just two basic examples but it is possible to search in many other ways:

-mtime	search files by modifying date
-atime	search files by last access date
-size	search files by file size
-user	search files by user they belong to

You need to be careful with quoting when using wildcards.

The wildcard * symbol represents a string of any character and of any length.

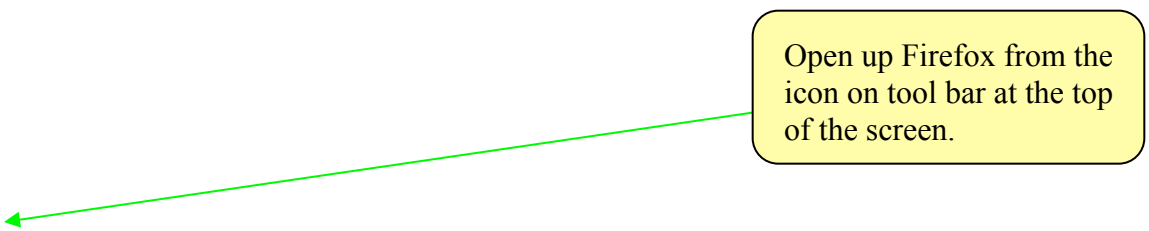
Obtaining and transferring information

The first step in exploiting genome sequences is obtaining your genome sequence. As time goes by there are more and more genome sequences available, from an ever increasing number of locations. Typically a complete genome sequence project is quite large, and therefore the files containing the data are going to be quite unwieldy. One of the simplest ways in which such information can be obtained is using **ftp** or 'file transfer protocol'. This a method of transferring information from a remote machine to the computer you are working on.

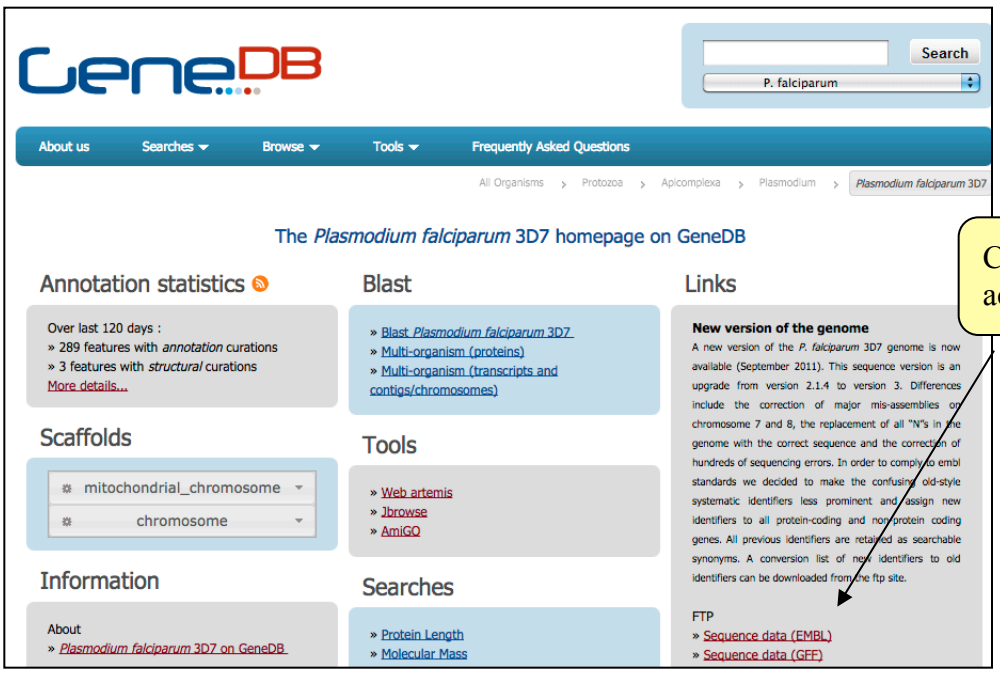
The **ftp** command can be used in UNIX to connect to a remote machine specified in the command line. Once a connection is established it is possible to both send (upload) and receive (download) data. However as we are limited for time we will not use this method, and instead use a more user-friendly method.

Using ftp on the internet

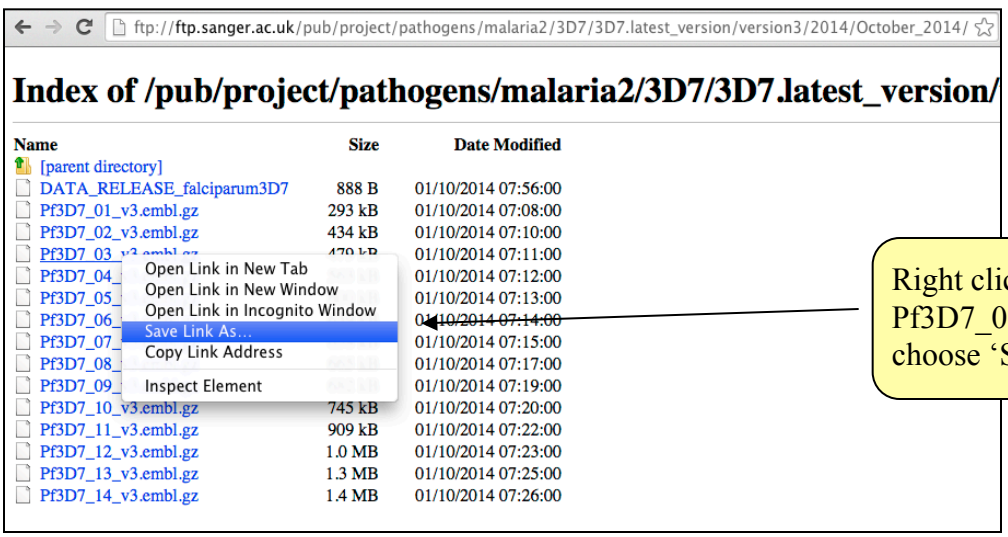
In addition to UNIX, ftp is also available on most Macs and PCs and allows you to transfer files readily between different computers worldwide. It is worth learning how to use **ftp**; most machines will have a graphical **ftp** interface and this makes file transfer very easy. Unfortunately there are a large number of alternatives and we can't show them all to you. Instead, we'll use **ftp** to download information to the current working directory on the computer you are working using the Firefox web browser. You are now going to an **ftp** web page where you are going to download the DNA sequence for *P. falciparum* chromosome 3.



In the location box type in the address **http://www.genedb.org/Homepage/Pfalciparum** and press return. The page below should appear.



Click on the ftp access link

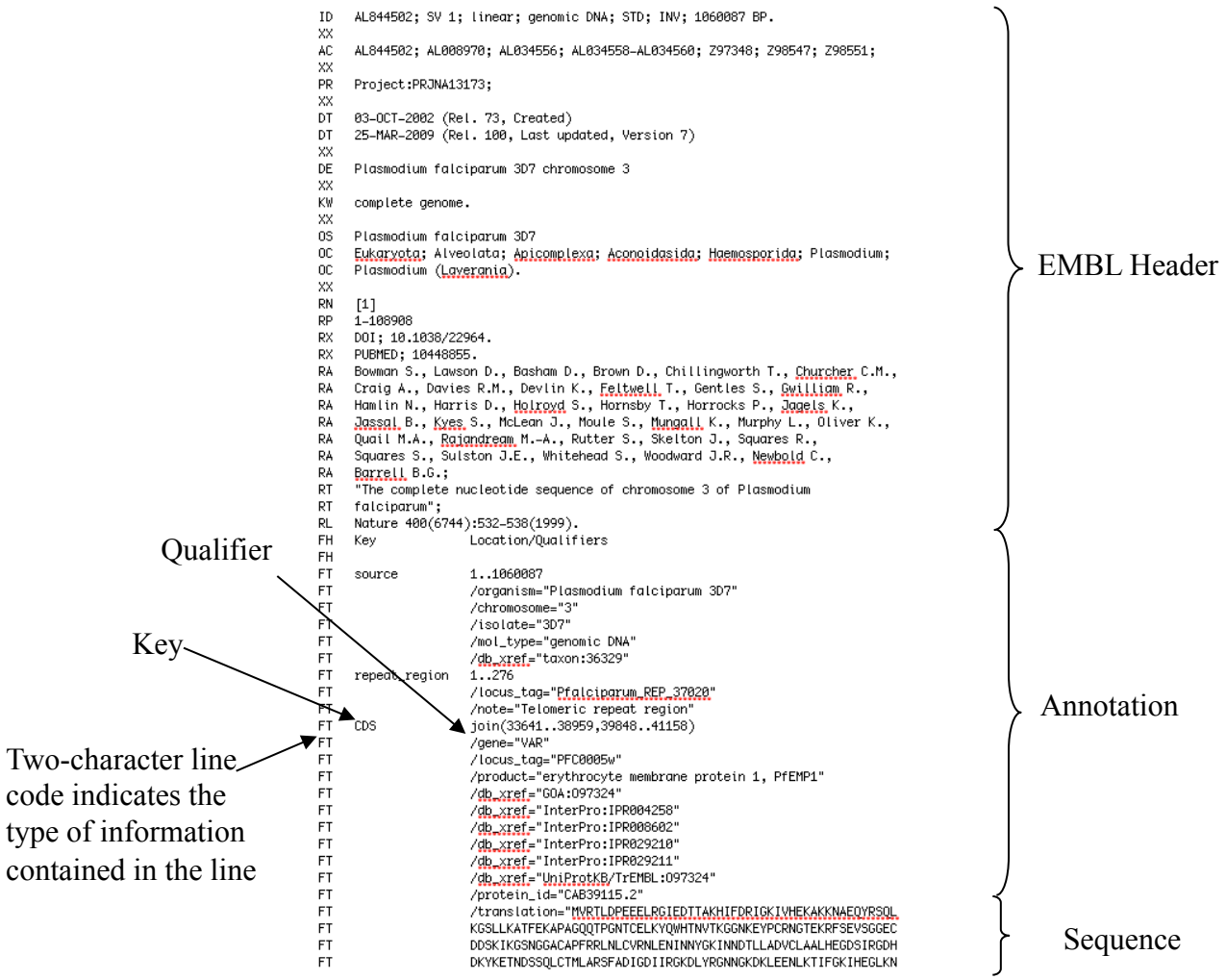


Right click on Pf3D7_03_v3.embl.gz and choose ‘Save Link As’.

Database Entries

The *P. falciparum* chr3 embl file can be obtained from the EMBL database at the European Bioinformatics Institute (http://www.ebi.ac.uk/) and is presented in a specific format with a series of defined qualifiers and keys (see below and Appendix IV) to help identify the different components of an entry.

Below is an example of a small EMBL entry with the different features of the entry highlighted




```
ID   AL844502; SV 1; linear; genomic DNA; STD; INV; 1060087 BP.
XX
AC   AL844502; AL008970; AL034556; AL034558-AL034560; Z97348; Z98547; Z98551;
XX
PR   Project:PRJNA13173;
XX
DT   03-OCT-2002 (Rel. 73, Created)
DT   25-MAR-2009 (Rel. 100, Last updated, Version 7)
XX
DE   Plasmodium falciparum 3D7 chromosome 3
XX
KW   complete genome.
XX
OS   Plasmodium falciparum 3D7
OC   Eukaryota; Alveolata; Apicomplexa; Aconoidasida; Haemosporida; Plasmodium;
OC   Plasmodium (Laverania).
XX
RN   [1]
RP   1-108908
RX   DOI; 10.1038/22964.
RX   PUBMED; 10448855.
RA   Bowman S., Lawson D., Basham D., Brown D., Chillingworth T., Churcher C.M.,
RA   Craig A., Davies R.M., Devlin K., Feltwell T., Gentles S., Gwilliam R.,
RA   Hamlin N., Harris D., Holroyd S., Hornsby T., Horrocks P., Jagels K.,
RA   Jassal B., Kyes S., McLean J., Moule S., Mungall K., Murphy L., Oliver K.,
RA   Quail M.A., Rajandream M.-A., Rutter S., Skelton J., Squares R.,
RA   Squares S., Sulston J.E., Whitehead S., Woodward J.R., Newbold C.,
RA   Barrell B.G.;
RT   "The complete nucleotide sequence of chromosome 3 of Plasmodium
RT   falciparum";
RL   Nature 400(6744):532-538(1999).
FH   Key             Location/Qualifiers
FH
FT   source          1..1060087
FT                   /organism="Plasmodium falciparum 3D7"
FT                   /chromosome="3"
FT                   /isolate="3D7"
FT                   /mol_type="genomic DNA"
FT                   /db_xref="taxon:36329"
FT   repeat_region   1..276
FT                   /locus_tag="Pfalciparum_REP_37020"
FT                   /note="Telomeric repeat region"
FT   CDS             join(33641..38959,39848..41158)
FT                   /gene="VAR"
FT                   /locus_tag="PFC0005w"
FT                   /product="erythrocyte membrane protein 1, PfEMP1"
FT                   /db_xref="GOA:Q97324"
FT                   /db_xref="InterPro:IPR004258"
FT                   /db_xref="InterPro:IPR008602"
FT                   /db_xref="InterPro:IPR029210"
FT                   /db_xref="InterPro:IPR029211"
FT                   /db_xref="UniProtKB/TrEMBL:Q97324"
FT                   /protein_id="CAB39115.2"
FT                   /translation="MVRTLDPEEELRGIEDTTAKHIFDRIGKIVHEKAKKNAEQYRSQL
FT                   KGSLLKATFEKAPAGQQTPGNTCELKYQWHTNVTKGGNKEYPCRNGTEKRFSEVSGGEC
FT                   DDSKIKGSNGGACAPFRRLNLCVRNLENINNYGKINNDTLLADVCLAALHEGDSIRGDH
FT                   DKYKETNDSSQLCTMLARSFADIGDIIRGKDLYRGNNGKDKLEENLKTIFGKIHEGLKN
```

In addition to the EMBL database, there are the mirror databases, Genbank (NCBI) and DDBJ (National Institute of Genetics, Japan), which contain the same sequence entries, but have slight differences in the way in which the information is presented. The next two pages contain the text of the complete entries for same sequence from the EMBL and GenBank databases, compare the two entries and identify the differences.

EMBL Entry

```
ID   ECRSMA     standard; DNA; PRO; 500 BP.
XX
AC   L40173;
XX
SV   L40173.1
XX
DT   10-AUG-1995 (Rel. 44, Created)
DT   04-MAR-2000 (Rel. 63, Last updated, Version 4)
XX
DE   Erwinia carotovora repressor (rsmA) gene, complete cds.
XX
KW   repressor; rsmA gene.
XX
OS   Pectobacterium carotovorum
OC   Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacteriaceae;
OC   Pectobacterium.
XX
RN   [1]
RP   1-500
RA   Cui Y., Chatterjee A., Liu Y., Dumenyo C.K., Chatterjee A.K.;
RT   "Identification of a global repressor gene, rsmA, of Erwinia carotovora
RT   subsp. carotovora that controls extracellular enzymes,
RT   N-(3-oxohexanoyl)-L-homoserine lactone, and pathogenicity in soft-rotting
RT   Erwinia spp";
RL   J. Bacteriol. 177(17):0-0(1995).
XX
DR   GOA; Q47620; Q47620.
DR   SWISS-PROT; Q47620; CSRA_ERWCA.
XX
FH   Key             Location/Qualifiers
FH
FT   source          1..500
FT                   /db_xref="taxon:554"
FT                   /organism="Pectobacterium carotovorum"
FT                   /strain="71"
FT                   /sub_species="carotovora"
FT                   /gene="rsmA"
FT   CDS             246..431
FT                   /codon_start=1
FT                   /db_xref="GOA:Q47620"
FT                   /db_xref="SWISS-PROT:Q47620"
FT                   /note="putative"
FT                   /transl_table=11
FT                   /gene="rsmA"
FT                   /function="global repressor"
FT                   /protein_id="AAA74502.1"
FT                   /translation="MLILTRRVGETLIIGDEVTVTVLGVKGNQVRIGVNAPKEVSVHRE
FT                   EIYQRIQAEKSQPTSY"
XX
SQ   Sequence 500 BP; 140 A; 101 C; 120 G; 139 T; 0 other;
     ggatccggca agcaggatag aaagtgtgtt accttcagat attctgaagc tttacatgct        60
     cagttctgtt gttgtgataa caaaagcaca agctactgat atcgactaaa ctaacaagta       120
     gtgacaaacc ggagtgtgat ggtgtggtta taccatcgtc taggtttacg ttttcacagc       180
     acatgatgga taatggcggg gagacagaga gacccgactc tttataatct ttcaaggagc       240
     aaagaatgct tattttgact cgtcgagttg gcgaaaccct catcatcggc gatgaggtaa       300
     cggttaccgt attaggagtg aaaggcaacc aggtgcgtat tggtgttaat gcacctaaag       360
     aggtttctgt ccaccgtgaa gagatctatc agcgtattca ggccgaaaaa tctcaaccaa       420
     cgtcatattg attgacaatg cgtctcgtgt tcgcgggacg caattgttat ttccggtttt       480
     tcccccacac atttatcgat                                                  500
//
```

GenBank Entry

```
LOCUS       ERWRSMA                  500 bp    DNA     linear   BCT 19-AUG-1995
DEFINITION  Erwinia carotovora repressor (rsmA) gene, complete cds.
ACCESSION   L40173
VERSION     L40173.1  GI:927031
KEYWORDS    repressor; rsmA gene.
SOURCE      Pectobacterium carotovorum
  ORGANISM  Pectobacterium carotovorum
            Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacteriaceae;
            Pectobacterium.
REFERENCE   1  (bases 1 to 500)
  AUTHORS   Cui,Y., Chatterjee,A., Liu,Y., Dumenyo,C.K. and Chatterjee,A.K.
  TITLE     Identification of a global repressor gene, rsmA, of Erwinia
            carotovora subsp. carotovora that controls extracellular enzymes,
            N-(3-oxohexanoyl)-L-homoserine lactone, and pathogenicity in
            soft-rotting Erwinia spp
  JOURNAL   J. Bacteriol. 177(17) (1995) In press
COMMENT     Original source text: Erwinia carotovora (strain 71, sub_species
            carotovora) DNA.
FEATURES             Location/Qualifiers
     source          1..500
                     /organism="Pectobacterium carotovorum"
                     /strain="71"
                     /sub_species="carotovora"
                     /db_xref="taxon:554"
     gene            107..431
                     /gene="rsmA"
     -10_signal      107..112
                     /gene="rsmA"
     RBS             235..239
                     /gene="rsmA"
     CDS             246..431
                     /gene="rsmA"
                     /function="global repressor"
                     /note="putative"
                     /codon_start=1
                     /transl_table=11
                     /protein_id="AAA74502.1"
                     /db_xref="GI:927032"
                     /translation="MLILTRRVGETLIIGDEVTVTVLGVKGNQVRIGVNAPKEVSVHR
                     EEIYQRIQAEKSQPTSY"
BASE COUNT      140 a    101 c    120 g    139 t
ORIGIN
        1 ggatccggca agcaggatag aaagtgtgtt accttcagat attctgaagc tttacatgct
       61 cagttctgtt gttgtgataa caaaagcaca agctactgat atcgactaaa ctaacaagta
      121 gtgacaaacc ggagtgtgat ggtgtggtta taccatcgtc taggtttacg ttttcacagc
      181 acatgatgga taatggcggg gagacagaga gacccgactc tttataatct ttcaaggagc
      241 aaagaatgct tattttgact cgtcgagttg gcgaaaccct catcatcggc gatgaggtaa
      301 cggttaccgt attaggagtg aaaggcaacc aggtgcgtat tggtgttaat gcacctaaag
      361 aggtttctgt ccaccgtgaa gagatctatc agcgtattca ggccgaaaaa tctcaaccaa
      421 cgtcatattg attgacaatg cgtctcgtgt tcgcgggacg caattgttat ttccggtttt
      481 tcccccacac atttatcgat

//
```

The two entries shown above contain the same biological information but differ in the format and presentation of this information. One of the most obvious difference is in the header region of the file that gives the background information to the submitted sequence. Another clear difference is that the EMBL entry has an additional **two letter line code** on the left hand margin.

EMBL entries are structured so as to be usable by human readers as well as by computer programs. The explanations, descriptions, classifications and other comments are in ordinary English for readability. At the same time, the structure is systematic enough to allow computer programs to easily read, identify, and manipulate the various types of data included.

Each line begins with a **two letter line code**, which indicates the type of information contained in the line. The currently used line types, along with their respective line codes, are listed below.

```
ID - identification             (begins each entry; 1 per entry)
AC - accession number           (>=1 per entry)
SV - new sequence identifier    (>=1 per entry)
DT - date                       (2 per entry)
DE - description                (>=1 per entry)
KW - keyword                    (>=1 per entry)
OS - organism species           (>=1 per entry)
OC - organism classification    (>=1 per entry)
OG - organelle                  (0 or 1 per entry)
RN - reference number           (>=1 per entry)
RC - reference comment          (>=0 per entry)
RP - reference positions        (>=1 per entry)
RX - reference cross-reference  (>=0 per entry)
RA - reference author(s)        (>=1 per entry)
RT - reference title            (>=1 per entry)
RL - reference location         (>=1 per entry)
DR - database cross-reference   (>=0 per entry)
FH - feature table header       (0 or 2 per entry)
FT - feature table data         (>=0 per entry)
CC - comments or notes          (>=0 per entry)
XX - spacer line                (many per entry)
SQ - sequence header            (1 per entry)
bb - (blanks) sequence data     (>=1 per entry)
// - termination line           (ends each entry; 1 per entry)
```