

R. Wünschiers

Computational Biology

Unix/Linux,
Data Processing and
Programming



Springer

Röbbe Wünschiers

Computational Biology – Unix/Linux, Data Processing and Programming

With 19 Figures and 12 Tables



Springer

Dr. Röbbel Wünschiers
University of Cologne
Institute for Genetics
Weyertal 121
50931 Köln
Germany

ISBN 3-540-21142-X Springer-Verlag Berlin Heidelberg New York

Library of Congress Control Number: 2004102411

This work is subject to copyright. All rights reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag is a part of Springer Science + Business Media
springeronline.com

Springer-Verlag Berlin Heidelberg 2004
Printed in Germany

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Cover design: Design & Production, Heidelberg
Typesetting: Camera ready by the author
31/3150WI - 5 4 3 2 1 0 - Printed on acid-free paper

Röbbe Wünschiers

*Computational Biology -
Unix/Linux, Data Processing and Programming*

Springer

Berlin

Heidelberg

New York

Hong Kong

London

Milan

Paris

Tokyo

Dedicated

to

Károly Nagy and to the Open-Source Community

Foreword

A shift in culture

Only a decade ago, the first thing a molecular biologist would have had to learn when he or she started the lab work was how to handle pipettes, extract DNA, use enzymes and clone a gene. Now, the first thing that he or she should learn is how to handle databases and to extract all the information that is already known about the gene that he or she wants to study. In all likelihood, he or she will find that the gene has already been sequenced from several organisms, that it was recovered in a variety of EST projects, that expression data are available from microarray and SAGE studies, that it was included in linkage studies, that proteomics data are rapidly accumulating, that lists of interacting proteins are being compiled, that domain structure data are available and that it is part of a network of genetic interactions which is intensively modelled. He or she will discover that all this information resides in many different databases with different data formats and with different levels of analyses and linking. Starting to work on this gene will make sense only, if all this information is put together in a project-specific manner and set into the context of what is known about related genes and processes. At this point he or she may decide to walk up to the bioinformatics group in house and ask for help with arranging the data in a useful manner. This will then turn into the first major frustration in his or her career, since the last thing a scientific bioinformatics group wants to do is to provide a service for data retrieval and management.

Molecular biology is currently going through a dramatic cultural shift. The daily business of pipetting and gel running has increasingly to be complemented with data compiling and processing. Large lists of data are produced by sequencers, microarray experiments or real-time PCR machines every day. Working with data lists has become as important as extracting DNA. Every bench scientist needs proficiency in computing; discoveries are made both at the bench and on the screen. It would be completely wrong to think that computers in molecular biology are the business of bioinformaticians only.

Bioinformatics has become a scientific discipline of its own and should not be considered to be a service provider. The day-to-day computing will always have to be done by the experimentalist himself or herself.

Of course, there are now also a lot of helpful and fancy program packages for the bench scientist; but these will only perform routine tasks and all too often they are only poorly compatible. A scientist needs the freedom to develop his or her own ideas and to link things that have previously not been linked. Being able to go back to the basics of computing and programming is therefore a vital skill for the experimentalist, as important as making buffers and setting up enzyme reactions. It allows him or her to handle and analyze the data in exactly the way it is required for the project and to pursue new avenues of research, rather than trotting old paths.

Unix is the key to basic computing. If one is used to Windows or Mac operating systems, this might at first sound like going back into the stone age; but the dramatic recent shift of at least the Mac operating system to a Unix base should teach us otherwise. Unix is here to stay and it allows the largest flexibility for bioinformatics applications. Those who have learned Unix will soon discover the myriad of little “progies” that are available from colleagues all over the world and that can make life much easier in the lab.

This book gives exactly the sort of introduction into Unix, Unix-based operating systems and programming languages that will be a key competence for experimentally working molecular biologists and that will make all the difference for the successful projects of the future. It has been written by a bench scientist, specifically with the needs of molecular biologists in mind. It can be used either for self-teaching or in practical courses. Every group leader should hand over this book to new students in the lab, together with their first set of pipettes.

Cologne/Germany,
January 2004

Prof. Dr. Diethard Tautz
(Institut für Genetik der Universität zu Köln)

Preface

Welcome on board!

With this book I would like to invite you, the scientist, to a journey through terminals and program codes. You are welcome to put aside your pipette, culture flask or rubber boots for a while, make yourself comfortable in front of a computer (do not forget your favourite hot alcohol-free drink) and learn some unixing and programming. *Why?* Because we are living in the information age and there is a huge amount of biological knowledge and databases out there. They contain information on almost everything: genes and genomes, rRNAs, enzymes, protein structures, DNA-microarray experiments, single organisms, ecological data, the tree of life and endless more. Furthermore, nowadays many research apparatuses are connected to computers. Thus, you have electronic access to your data. However, in order to cope with all this information you need some tools. This book will provide you with the skills to use these tools and to develop your own tools, i.e. it will introduce Unix and its derivatives (Linux, Mac OS X, CygWin, etc.) and programming (shell programming, awk, perl). These tools will make you independent of the way in which other people make you process your data – in the form of application software. What you want is open functionality. You want to decide how to process (e.g. analyze, format, save, correlate) data and you want it now – not waiting for the lab programmer to treat your request; and you know it best – you understand your data and your demands. This is what open functionality stands for, and both Linux and programming languages can provide it to you.

I started programming on a Casio PB-100 hand-held built in 1983. It can store 10 small Basic programs. The accompanying book was entitled “Learn as you go” and, indeed, in my opinion this is the best way to learn programming. My first contact to Unix was triggered by the need to copy data files from a Unix-driven Bruker EPR-Spectrometer onto a floppy disk. The real challenge started when I tried to import the files to a data-plotting program on the PC. While the first problem could be solved by finding the right page in a Unix manual, the latter required programming skills – Q-Basic at that time. This

problem was minor compared to the trouble one encounters today. A common problem is to feed one program with the output of another program: you might have to change lines to columns, commas to dots, tabulators to semicolons, uppercase to lowercase, DNA to RNA, FASTA to GenBank format and so forth. Then there is that huge amount of information out there in the web, which you might need to bring into shape for your own analysis.

You and This Book – This book is written for the total beginner. You need not even to know what a computer is, though you should have access to one and find the power switch. The book is the result of a) the way how I learned to work with Unix, its derivatives and its numerous tools and b) a lecture which I started at the Institute for Genetics at the University of Cologne/Germany. Most programming examples are taken from biology; however, you need not be a biologist. Except for two or three examples, no biological knowledge is necessary. I have tried to illustrate almost everything practically with so-called terminals and examples. You should run these examples. Each chapter closes with some exercises. Brief solutions can be found at the end of the book.

Why Linux? – This book is not limited to Linux! All examples are valid for Unix or any Unix derivative like *Mac OS X*, *Knoppix* or the free Windows-based *CygWin* package, too. I chose Linux because it is open source software: you need not invest money except for the book itself. Furthermore, Linux provides all the great tools Unix provides. With Linux (as with all other Unix derivatives) you are close to your data. Via the command line you have immediate access to your files and can use either publicly available or your own designed tools to process these. With the aid of *pipes* you can construct your own data-processing pipeline. It is great.

Why awk and perl? – **awk** is a great language for both learning programming and treating large text-based data files (contrary to binary files). To 99% you will work with text-based files, be it data tables, genomes or species lists. Apart from being simple to learn and having a clear syntax, **awk** provides you with the possibility to construct your own commands. Thus, the language can grow with you as you grow with the language. I know bioinformatic professionals entirely focusing on **awk**. **perl** is much more powerful but also more unclear in its syntax (or flexible, to put it positively), but, since **awk** was one basis for developing **perl**, it is only a small step to go once you have learned **awk** – but a giant leap for your possibilities. You should take this step. By the way, both **awk** and **perl** run on all common operating systems.

Acknowledgements – Special thanks to Kristina Auerswald, Till Bayer, Benedikt Bosbach and Chris Voolstra for proofreading, and all the other students for encouraging me to bring these lines together.

Contents

Part I Whetting Your Appetite

1	Introduction	3
1.1	Information	3
1.2	What Is Linux?	3
1.3	What Is Shell Programming?	4
1.4	What Is <code>sed</code> ?	4
1.5	What Is <code>awk</code> ?	4
1.6	What Is <code>perl</code> ?	5
1.7	Prerequisites	5
1.8	Conventions	6

Part II Computer and Operating Systems

2	Unix/Linux	9
2.1	What Is a Computer?	9
2.2	Some History	10
2.2.1	Versions of Unix	11
2.2.2	The Rise of Linux	12
2.2.3	Why a Penguin?	14
2.2.4	Linux Distributions	14
2.3	X-Windows	14
2.3.1	How Does It Work?	15
2.4	Unix/Linux Architecture	16
2.5	What Is the Difference Between Unix and Linux?	17
2.6	What Is the Difference Between Unix/Linux and Windows?	17
2.7	What Is the Difference Between Unix/Linux and Mac OS X?	18
2.8	One Computer, Two Operating Systems	18
2.8.1	VMware	19
2.8.2	CygWin	19

2.8.3	Wine	19
2.8.4	Others	20
2.9	Knoppix	20
2.10	Software Running Under Linux	21
2.10.1	Biosciences	21
2.10.2	Office and Co.	22

Part III Working with Unix/Linux

3	The First Touch	27
3.1	Login	27
3.1.1	Working with CygWin or Mac OS X	27
3.1.2	Working Directly on a Unix/Linux Computer	28
3.1.3	Working on a Remote Unix/Linux Computer	28
3.2	Using Commands	30
3.2.1	Syntax of Commands	31
3.2.2	Editing the Command Line	32
3.2.3	Change Password	32
3.2.4	How to Find out More	33
3.3	Logout	33
	Exercises	34
4	Working with Files	35
4.1	Browsing Files and Directories	36
4.2	File Attributes	38
4.3	Special Files: . and	39
4.4	Protecting Files and Directories	41
4.4.1	Directories	41
4.4.2	Files	41
4.4.3	Examples	41
4.4.4	Changing File Attributes	42
4.4.5	Extended File Attributes	44
4.5	File Archives	44
4.6	File Compression	47
4.7	Searching for Files	49
	Exercises	50
5	Installing BLAST and ClustalW	53
5.1	Downloading the Programs via FTP	53
5.1.1	Downloading BLAST	54
5.1.2	Downloading ClustalW	55
5.2	Installing BLAST	57
5.3	Running BLAST	57
5.4	Installing ClustalW	59

5.5	Running ClustalW	59
	Exercises	60
6	Working with Text	63
6.1	A Quick Start: <code>cat</code>	64
6.1.1	Text Sorting	65
6.1.2	View File Beginning or End	67
6.1.3	Scrolling Through Files	67
6.1.4	Character, Word and Line Counting	67
6.1.5	Finding Text	68
6.1.6	Text File Comparisons	69
6.2	<code>pico</code>	70
6.3	<code>vi</code> and <code>vim</code>	72
6.3.1	Immediate Takeoff	72
6.3.2	Starting <code>vi</code>	73
6.3.3	Modes	73
6.3.4	Moving the Cursor	74
6.3.5	Doing Corrections	75
6.3.6	Save and Quit	76
6.3.7	Copy and Paste	76
6.3.8	Search and Replace	78
	Exercises	78
7	Using the Shell	81
7.1	What Is the Shell?	81
7.2	Different Shells	82
7.3	Setting the Default Shell	83
7.4	Useful Shortcuts	84
7.5	Redirections	85
7.6	Pipes	87
7.7	Lists of Commands	88
7.8	Aliases	89
7.9	Scheduling Commands	89
7.10	Wildcards	91
7.11	Processes	92
7.11.1	Checking Processes	93
7.11.2	Background Processes	94
7.11.3	Killing Processes	95
	Exercises	96
8	Shell Programming	97
8.1	Script Files	97
8.2	Modifying the Path	99
8.3	Variables	100
8.4	Input and Output	102

8.4.1	<code>echo</code>	103
8.4.2	Here Documents: <code><<</code>	103
8.4.3	<code>read</code> and <code>line</code>	104
8.4.4	Script Parameters	105
8.5	Substitutions and Expansions	106
8.5.1	Variable Substitution	107
8.5.2	Command Expansion	108
8.6	Quoting	109
8.6.1	Escape Character	109
8.6.2	Single Quotes	109
8.6.3	Double Quotes	109
8.7	Decisions – Flow Control	110
8.7.1	<code>if...then...elif...else...fi</code>	110
8.7.2	<code>test</code>	111
8.7.3	<code>while...do...done</code>	113
8.7.4	<code>until...do...done</code>	115
8.7.5	<code>for...in...do...done</code>	116
8.7.6	<code>case...in...esac</code>	117
8.7.7	<code>select...in...do</code>	118
8.8	Debugging	119
8.8.1	<code>bash -xv</code>	120
8.8.2	<code>trap</code>	121
8.9	Examples	122
8.9.1	Check for DNA as File Content	122
8.9.2	Time Signal	123
8.9.3	Select Files to Archive	123
8.9.4	Remove Spaces	125
	Exercises	125
9	Regular Expressions	127
9.1	Using Regular Expressions	128
9.2	Search Pattern and Examples	130
9.2.1	Single-Character Meta Characters	130
9.2.2	Quantifiers	133
9.2.3	Grouping	134
9.2.4	Anchors	134
9.2.5	Escape Sequences	135
9.2.6	Alternation	136
9.2.7	Back References	137
9.2.8	Character Classes	138
9.2.9	Priorities	138
9.2.10	<code>egrep</code> Options	139
9.3	Regular Expressions and <code>vim</code>	139
	Exercises	140

10	Sed	141
10.1	When to Use sed ?	142
10.2	Getting Started	142
10.3	How sed Works	144
10.3.1	Pattern Space	144
10.3.2	Hold Space	145
10.4	sed Syntax	145
10.4.1	Addresses	146
10.4.2	sed and Regular Expressions	147
10.5	Commands	148
10.5.1	Substitutions	149
10.5.2	Transliterations	151
10.5.3	Deletions	152
10.5.4	Insertions and Changes	152
10.5.5	sed Script Files	154
10.5.6	Printing	155
10.5.7	Reading and Writing Files	156
10.5.8	Advanced sed	157
10.6	Examples	157
10.6.1	Gene Tree	157
10.6.2	File Tree	158
10.6.3	Reversing Line Order	159
	Exercises	160

Part IV Programming

11	Awk	163
11.1	Getting Started	164
11.2	awk 's Syntax	165
11.3	Example File	166
11.4	Patterns	167
11.4.1	Regular Expressions	167
11.4.2	Pattern-Matching Expressions	168
11.4.3	Relational Character Expressions	169
11.4.4	Relational Number Expressions	171
11.4.5	Mixing and Conversion of Numbers and Characters	171
11.4.6	Ranges	172
11.4.7	BEGIN and END	173
11.5	Variables	174
11.5.1	Assignment Operators	174
11.5.2	Increment and Decrement	176
11.5.3	Predefined Variables	177
11.5.4	Arrays	181
11.5.5	Shell Versus awk Variables	185

11.6	Scripts and Executables	185
11.7	Decisions – Flow Control	186
11.7.1	<code>if...else</code>	187
11.7.2	<code>while</code>	188
11.7.3	<code>do...while</code>	189
11.7.4	<code>for</code>	189
11.7.5	Leaving Loops	190
11.8	Actions	191
11.8.1	Printing	191
11.8.2	Numerical Calculations	195
11.8.3	String Manipulation	196
11.8.4	System Commands	200
11.8.5	User-Defined Functions	201
11.9	Input with <code>getline</code>	206
11.9.1	Reading from a File	206
11.9.2	Reading from the Shell	209
11.10	Examples	210
11.10.1	Sort an Array by Indices	210
11.10.2	Sum up Atom Positions	210
11.10.3	Translate DNA to Protein	211
11.10.4	Calculate Atomic Composition of Proteins	212
11.10.5	Calculate Distance Between Cysteines	215
	Exercises	220
12	Perl	221
12.1	Intention of this Chapter	221
12.2	Running <code>perl</code> Scripts	222
12.3	Variables	222
12.3.1	Scalars	223
12.3.2	Arrays	224
12.3.3	Hashes	229
12.3.4	Built-in Variables	233
12.4	Decisions – Flow Control	234
12.4.1	<code>if...elseif...else</code>	234
12.4.2	<code>unless</code> , <code>die</code> and <code>warn</code>	234
12.4.3	<code>while</code>	235
12.4.4	<code>do...while</code>	235
12.4.5	<code>until</code>	235
12.4.6	<code>do...until</code>	235
12.4.7	<code>for</code>	236
12.4.8	<code>foreach</code>	236
12.4.9	Controlling Loops	237
12.5	Data Input	239
12.5.1	Command Line	239
12.5.2	Internal	239

12.5.3	Files	240
12.6	Data Output	242
12.6.1	<code>print</code> , <code>printf</code> and <code>sprintf</code>	242
12.6.2	Here Documents: <code><<</code>	242
12.6.3	Files	243
12.7	Hash Databases	244
12.8	Regular Expressions	245
12.8.1	Special Escape Sequences	245
12.8.2	Matching: <code>m/.../</code>	246
12.9	String Manipulations	247
12.9.1	Substitute: <code>s/.../.../</code>	247
12.9.2	Transliterate: <code>tr/.../.../</code>	247
12.9.3	Common Commands	248
12.10	Calculations	249
12.11	Subroutines	250
12.12	Packages and Modules	251
12.13	Bioperl	254
12.14	You Want More?	254
12.15	Examples	254
12.15.1	Reverse Complement DNA	255
12.15.2	Calculate GC Content	255
12.15.3	Restriction Enzyme Digestion	256
12.15.4	Levenshtein Distance of Sequences	259
	Exercises	263
A	Appendix	265
A.1	Keyboard Shortcuts	265
A.2	Boot Problems	266
A.3	Optimizing the Bash Shell	266
A.3.1	Startup Files	266
A.3.2	A Helpful Shell Prompt	267
A.4	Text File Conversion (Unix ↔ DOS)	267
A.5	Devices	267
A.6	Mounting Filesystems	268
A.6.1	Floppy Disk	269
A.6.2	CD-ROM	270
A.6.3	PCMCIA and Compact Flash	270
A.7	Nucleotide and Protein Codes	271
A.8	Special Characters	272
	Solutions	273
	References	279
	Index	281