# System Programming

## Week 1: Introduction to System Programming

---

Seongjin Lee

Updated: 2016-09-07

01-intro

insight@hanyang.ac.kr

http://esos.hanyang.ac.kr

Esos Lab. Hanyang University

# Table of contents



# Nutshell

Ken Thompson and Dennis Ritchie at PDP-11 in 1971 (Photo: Courtesy of Bell Labs)

# What we are going to learn

- Familiarize with Unix
- Experience systems programming
- Understand fundamental OS concepts
  - Multi-user concepts
  - Basic and advanced I/O
  - Process
  - Interprocess communication

# Why do we have to?

- Unix gives you insights on how other OS works
- You can only catch the tiger by going into the tiger's den
- It is the basis for most other programming and understanding of the system
- It in C helps you understand the general programming concepts

# How are going to do?

```
1  /*
2   * welcome file
3   */
4
5  #include <stdio.h>
6  #include <unistd.h>
7
8  int
9  main(int argc, char **argv) {
10    printf("Welcome to System Programming, %s!\n", getlogin());
11 }
```

How to compile

$ cc -Wall -g -o welcome welcome.c

# Errors and Warnings

```
$ gcc -Wall -g -o welcome welcome.c
welcome.c:6:81: warning: implicit declaration of function 'getlogin'
        is invalid in C99 [-Wimplicit-function-declaration]
        printf("Welcome to System Programming, %s!\n", getlogin())
                                                       ^
welcome.c:6:81: warning: format specifies type 'char *' but the
        argument has type 'int' [-Wformat]
        printf("Welcome to System Programming, %s!\n", getlogin())
                                                    ~~      ^~~~~~~~~~
                                                    %d
welcome.c:6:92: error: expected ';' after expression
        printf("Welcome to System Programming, %s!\n", getlogin())
                                                                  ^
                                                                  ;
2 warnings and 1 error generated.
```

# About this class

Textbook

- "Advanced Programming in the UNIX Environment", by W. Richard Stevens, Stephen A. Rago (3rd Edition)

Assistant

- Yeonjin Noh (nyg0813@gmail.com)
- FTC #804

Grading:

- Attendance 5%
- Assignments 30%
- Quiz 5%
- Midterm Exam 30%
- Final Exam 30%
- Level test on editors

# Syllabus

Week 1 09-07 Introduction to system programming & Shell Survival Kit
Week 2 09-14 (추석)
Week 3 09-21 Files IO & ctag/etag
Week 4 09-28 Files and Directories
Week 5 10-05 Standard I/O Library
Week 6 10-12 Process Environment
Week 7 10-19 Process Control
Week 8 10-26 (중간고사)
Week 9 11-02 Signals
Week 10 11-09 Threads
Week 11 11-16 Thread Control
Week 12 11-23 Advanced I/O
Week 13 11-30 Interprocess Communication I
Week 14 12-07 Interprocess Communication II
Week 15 12-14 Network IPC Week 16 12-21 (기말고사)

# History

# The UNIX History

For more info : http://www.unix.org/what_is_unix/history_timeline.html

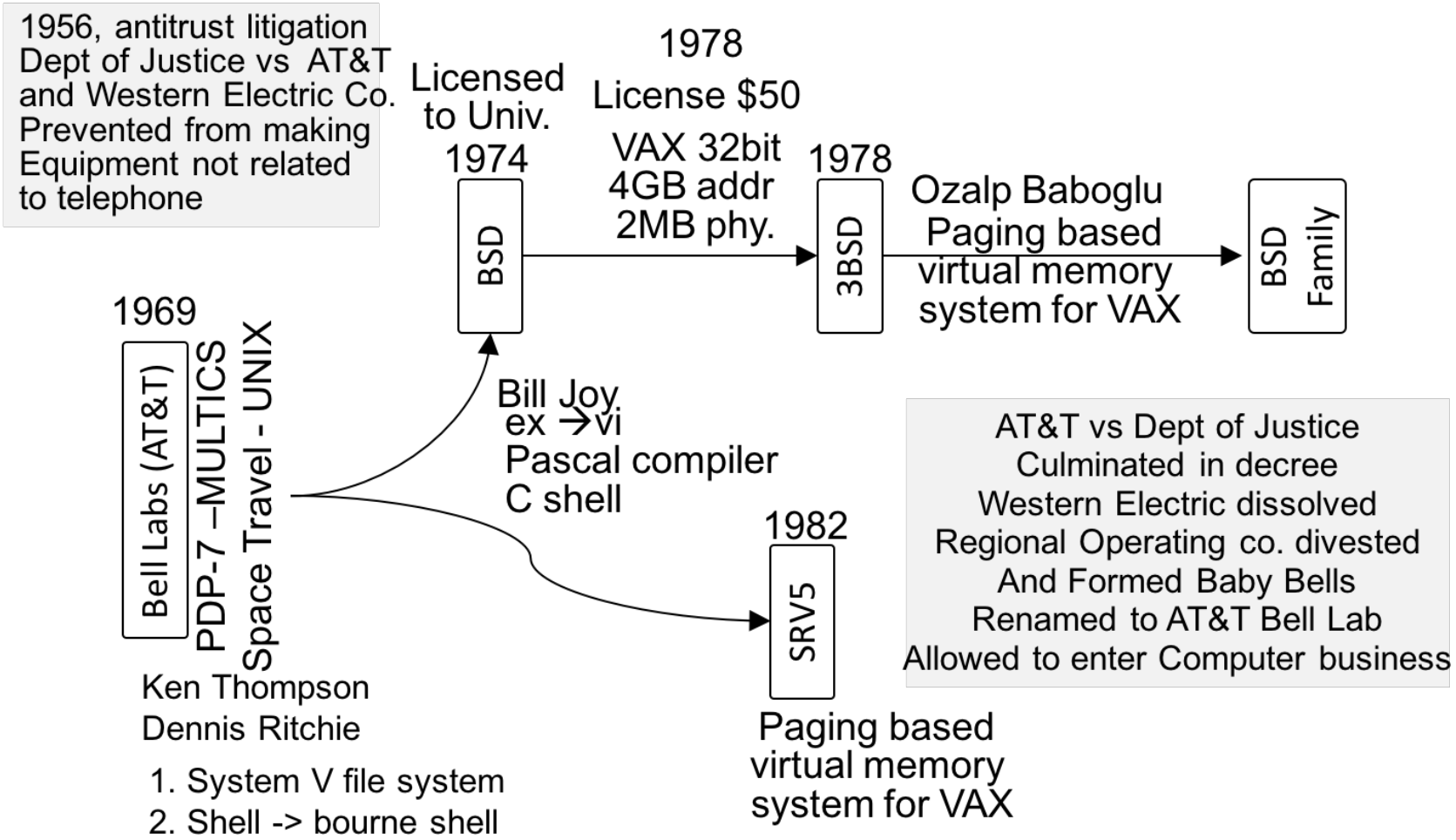
- Originally developed in 1969 at Bell Labs by Ken Thompson and Dennis Ritchie.
- 1973, Rewritten in C. This made it portable and changed the history of OS
- 1974: Thompson, Joy, Haley and students at Berkeley develop the **B**erkeley **S**oftware **D**istribution (BSD) of UNIX
- two main directions emerge: BSD and what was to become "System V"

# UNIX History

- 1984 4.2BSD released (TCP/IP)
- 1986 4.3BSD released (NFS)
- 1991 Linus Torvalds starts working on the Linux kernel
- 1993 Settlement of USL vs. BSDi; NetBSD, then FreeBSD are created
- 1994 Single UNIX Specification introduced
- 1995 4.4BSD-Lite Release 2 (last CSRG release); OpenBSD forked off NetBSD
- 2000 Darwin created (derived from NeXT, FreeBSD, NetBSD)
- 2003 Xen; SELinux
- 2005 Hadoop; DTrace; ZFS; Solaris Containers
- 2006 AWS ("Cloud Computing" comes full circle)
- 2007 iOS; KVM appears in Linux
- 2008 Android; Solaris open sourced as OpenSolaris

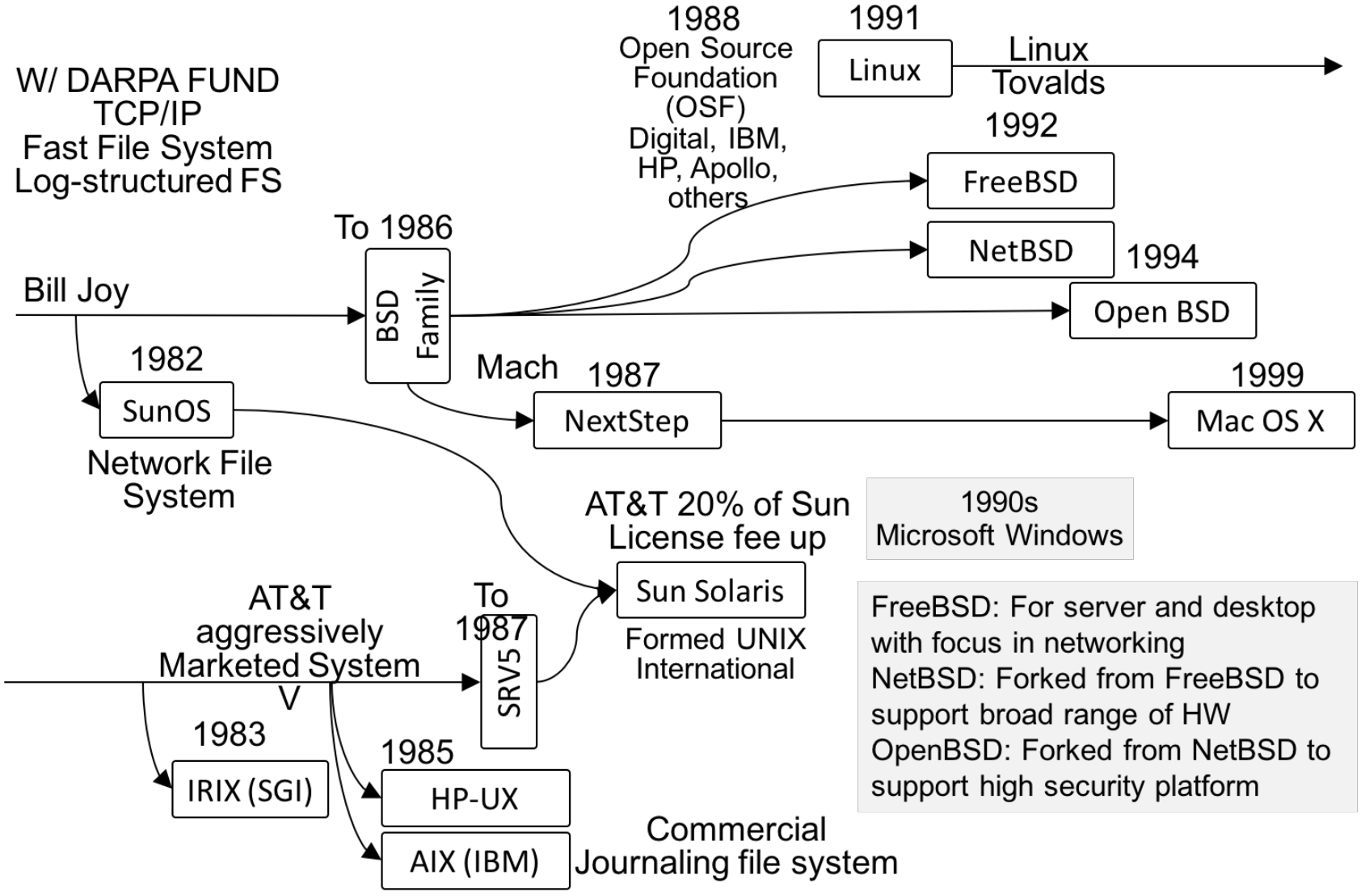
list from www.cs.stevens.edu/~jschauma/631A/

# History




1971 Nov. R&T – UNIX programmer's Manual
1973 Feb. UNIX v3 included cc – rewritten in C
1973 Oct. ACM SOSP – The UNIX Time Sharing System

# History


1988
Open Source
Foundation
(OSF)
Digital, IBM,
HP, Apollo,
others
1991
Linux
Linux
Tovalds
W/ DARPA FUND
TCP/IP
Fast File System
Log-structured FS
1992
FreeBSD
To 1986
BSD
Family
NetBSD
1994
Bill Joy
Open BSD
1982
SunOS
Mach
1987
NextStep
1999
Mac OS X
Network File
System
AT&T 20% of Sun
License fee up
1990s
Microsoft Windows
Sun Solaris
AT&T
aggressively
Marketed System
V
To
1987
SRV5
Formed UNIX
International
FreeBSD: For server and desktop
with focus in networking
NetBSD: Forked from FreeBSD to
support broad range of HW
OpenBSD: Forked from NetBSD to
support high security platform
1983
1985
IRIX (SGI)
HP-UX
AIX (IBM)
Commercial
Journaling file system

# Some UNIX Versions

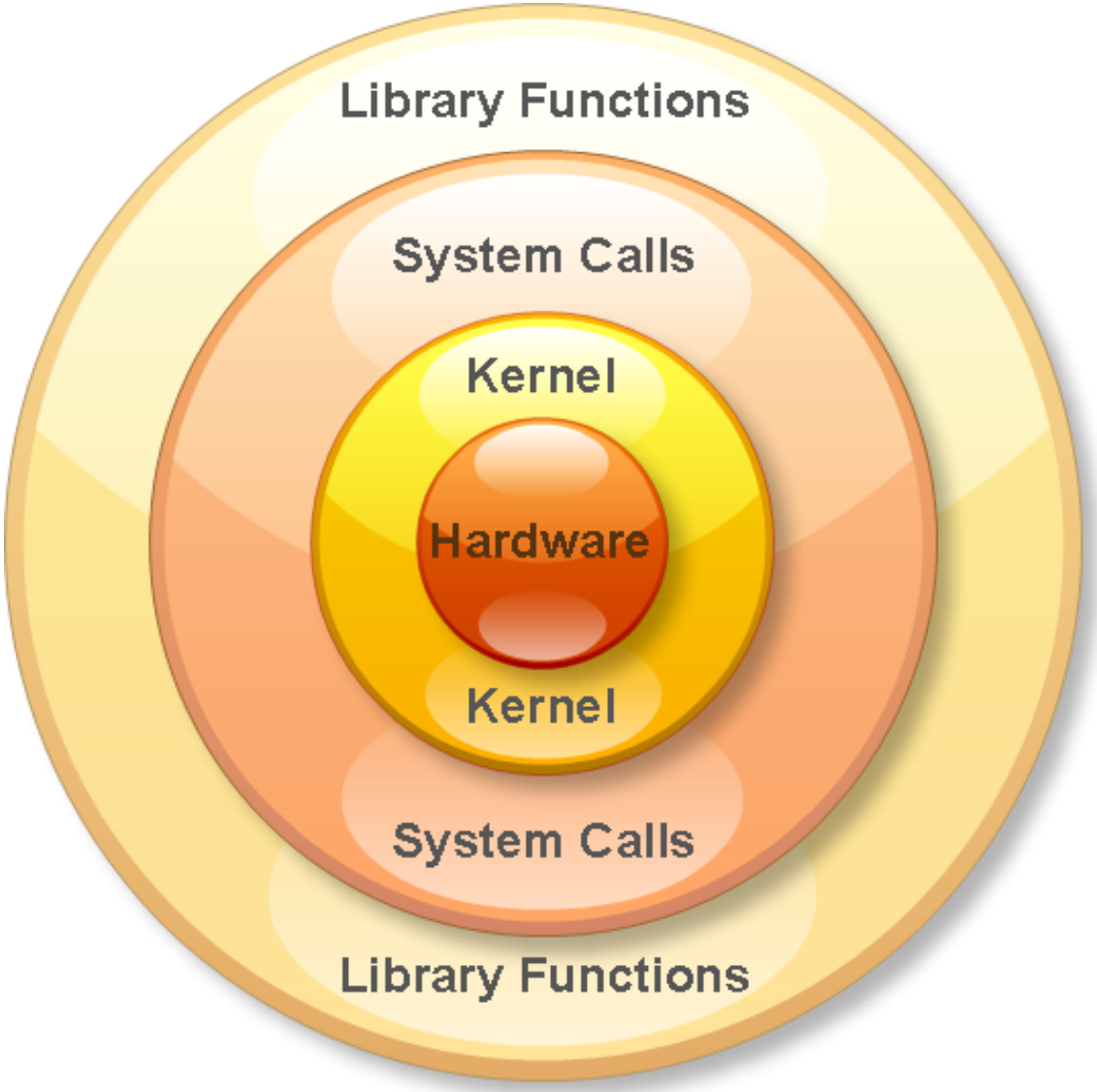
## More UNIX (some generic, some trademark, some just unix-like):

| | | | | |
|---|---|---|---|---|
| 1BSD | 2BSD | 3BSD | 4BSD | 4.4BSD Lite 1 |
| 4.4BSD Lite 2 | 386 BSD | A/UX | Acorn RISC iX | AIX |
| AIX PS/2 | AIX/370 | AIX/6000 | AIX/ESA | AIX/RT |
| AMiX | AOS Lite | AOS Reno | ArchBSD | ASV |
| Atari Unix | BOS | BRL Unix | BSD Net/1 | BSD Net/2 |
| BSD/386 | BSD/OS | CB Unix | Chorus | Chorus/MiX |
| Coherent | CTIX | Darwin | Debian GNU/Hurd | DEC OSF/1 ACP |
| Digital Unix | DragonFly BSD | Dynix | Dynix/ptx | ekkoBSD |
| FreeBSD | GNU | GNU-Darwin | HPBSD | HP-UX |
| HP-UX BLS | IBM AOS | IBM IX/370 | Interactive 386/ix | Interactive IS |
| IRIX | Linux | Lites | LSX | Mac OS X |
| Mac OS X Server | Mach | MERT | MicroBSD | Mini Unix |
| Minix | Minix-VMD | MIPS OS | MirBSD | Mk Linux |
| Monterey | more/BSD | mt Xinu | MVS/ESA OpenEdition | NetBSD |
| NeXTSTEP | NonStop-UX | Open Desktop | Open UNIX | OpenBSD |
| OpenServer | OPENSTEP | OS/390 OpenEdition | OS/390 Unix | OSF/1 |
| PC/IX | Plan 9 | PWB | PWB/UNIX | QNX |
| QNX RTOS | QNX/Neutrino | QUNIX | ReliantUnix | Rhapsody |
| RISC iX | RT | SCO UNIX | SCO UnixWare | SCO Xenix |
| SCO Xenix System V/386 | Security-Enhanced Linux | Sinix | Sinix ReliantUnix | Solaris |
| SPIX | SunOS | Tru64 Unix | Trusted IRIX/B | Trusted Solaris |
| Trusted Xenix | TS | UCLA Locus | UCLA Secure Unix | Ultrix |
| Ultrix 32M | Ultrix-11 | Unicos | Unicos/mk | Unicox-max |
| UNICS | UNIX 32V | UNIX Interactive | UNIX System III | UNIX System IV |
| UNIX System V | UNIX System V Release 2 | UNIX System V Release 3 | UNIX System V Release 4 | UNIX System V/286 |
| UNIX System V/386 | UNIX Time-Sharing System | UnixWare | UNSW | USG |
| Venix | Wollogong | Xenix OS | Xinu | xMach |

adopted from http://www.cs.stevens.edu/~jschauma/631A/

# The Unix Basics

# The UNIX Basics: Architecture




applications > shell

# System Calls and Library Calls

## System calls

They are entry points into kernel code where their functions are implemented. Documented in section 2 of the manual (e.g. `write(2)` or `$man 2 write`).

## Library Calls

They are transfers to user code which performs the desired functions. Documented in section 3 of the manual (e.g. `printf(3)`).

# Error Handling: ANSI C

- Important ANSI C Features:
  - function prototypes
  - generic pointers (`void *`)
  - abstract data types (e.g. `pid_t`, `size_t`)
- Error Handling:
  - meaningful return values
  - `errno` variable
  - look up constant error values via two functions:

```
1  #include <string.h>
2  char *strerror(int errnum) // returns pointer to message string
```

```
1  #include <stdio.h>
2  void perror(const char *msg)
```

# Principles

https://en.wikipedia.org/wiki/Unix_philosophy

- Small is beautiful.
- Make each program do one thing well.
- Build a prototype as soon as possible.
- Choose portability over efficiency.
- Store data in flat text files.
- Use software leverage to your advantage.
- Use shell scripts to increase leverage and portability.
- Avoid captive user interfaces.
- Make every program a filter.

# The Editors

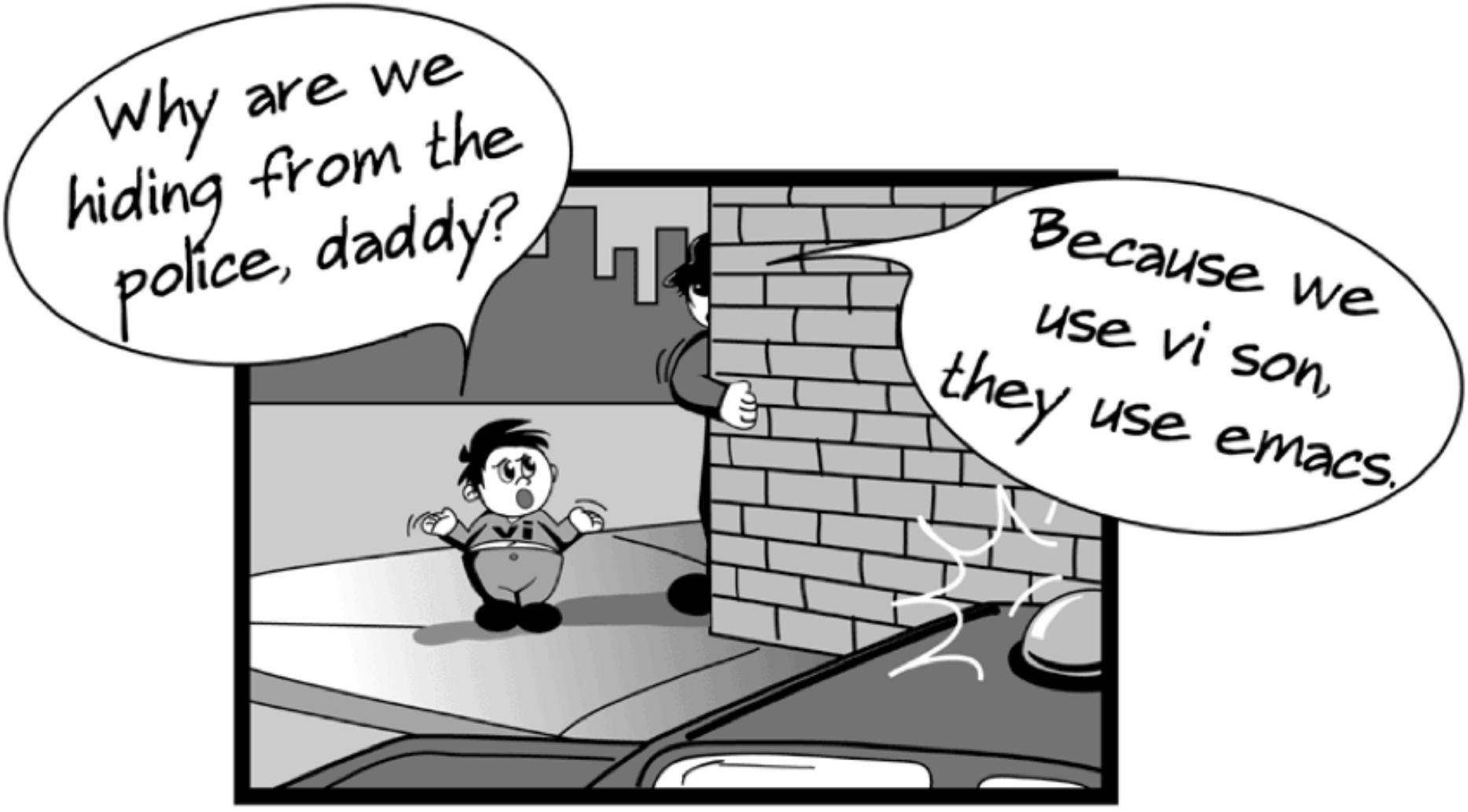
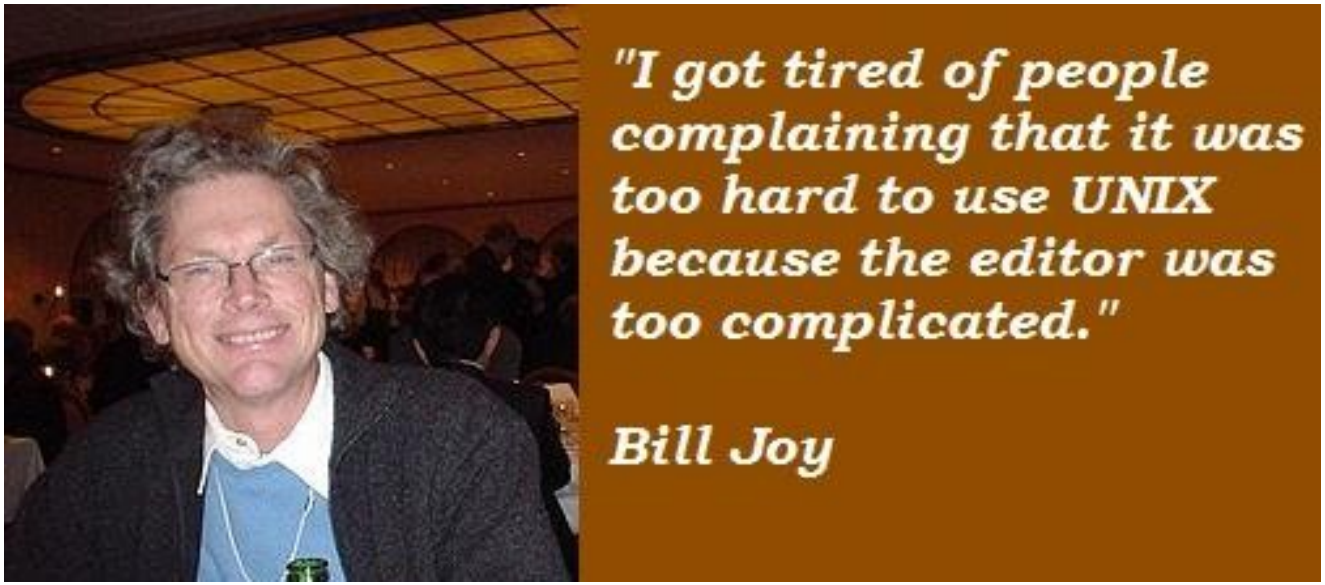



Image from http://michael-prokop.at/computer/tools_vim_en.html

# What is Vi(m)

**Vi** is a visual screen text editor developed by Bill Joy, who later becomes co-founder of Sun Micro Systems

- It is visual version of **ex**, a Unix line editor
- Vi is available on most Unix Systems
- Works with a variety of terminals
- Allows **ex** command from **vi**




# What is Vim

**VIM** is acronym for **Vi** i**M**proved, developed by Bram Moolenaar, a extended version of vi and some of enhancements include

- Completion, comparison, and merging of files
- Split and tabbed windows
- Command histories

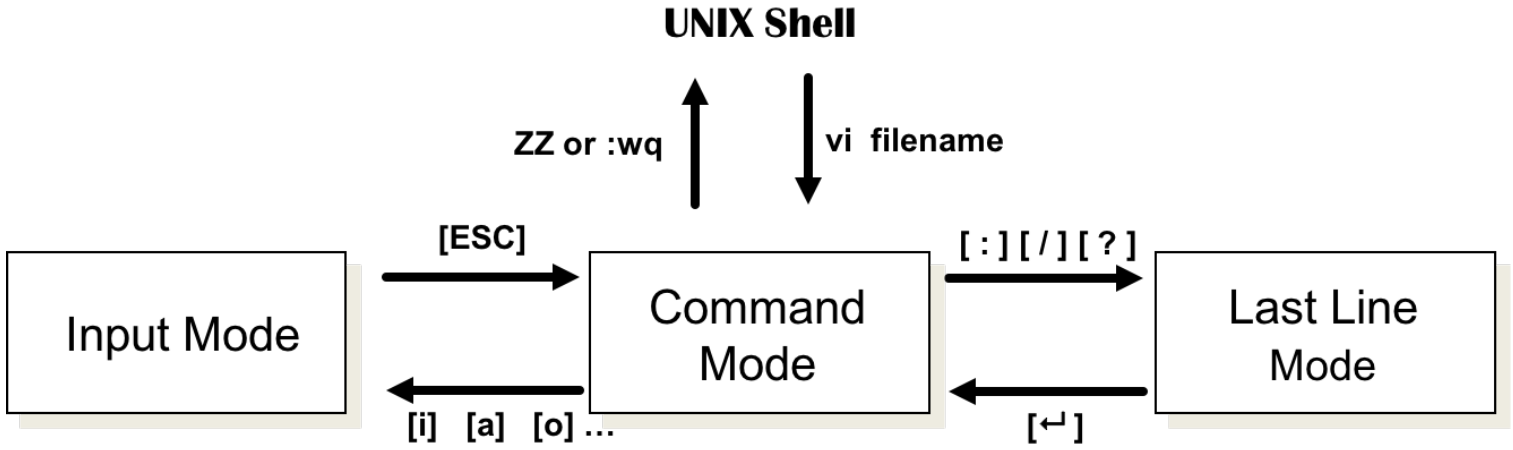

All editing session before saving is done in buffer area

- Nothing is saved as hard data, until you save it

# Modes of vi

There are three mode in vi

- **Command Mode** – A default mode in vi
  - Everything is command before you enter into other modes
- **Input Mode** – What you type is what you see
  - Anything typed in this mode is considered as data
  - Pressing [ESC] always leads to Command mode
- **Last Line Mode** – Only can be accessed from Command mode
  - Three ways to enter Last Line Mode – : (Colon) / (Back Slash) ? (Question Mark)




# Moving Around

VI uses four characters to move around, and each character is mapped to a direction


Moving by units of word, sentence, paragraph

- E.g., 3w moves to three words after the current cursor

`w` next word `b` previous word `(` Beginning of sentence

`)` end of sentence `{` Beginning of paragraph `}` end of paragraph

# Deleting

Deleting a character, words, sentence, line, and paragraph

- `x` erases a character
- Combination of direction commands with `d` erases a word, sentence, and paragraph.
  - E.g., `dw` erases a word before the cursor
- `dd` erases a line
- `D` to delete rest of line
- `X` to delete before the cursor
- `Xp` to transpose

# Searching and Replacing

Searching in vi is done in last line mode

- `/` lets you search a character, word, and words
  - E.g., `/abc` moves the cursor to the location of the pattern
- Search pattern in forward direction: `n`, backward direction: keystrokeN
- Regular expressions can be also used in searching

`r` replaces a character

- Suppose the cursor is on `b`, and by `r` `p` we can change it to "preview"

`breview` → `preview`

# Substitution

**Substituting in vi is done in last line mode**

Find i and substitute with X once

`:s/i/X` → This is preview. → ThXs is preview.
How it is done How it is done

Find i and substitute with X in the same line

`:s/i/X/g` → This is preview. → ThXs is prevXew.
How it is done How it is done

Find i and substitute with X in all the lines

`:%s/i/X/g` → This is preview. → ThXs is prevXew.
How it is done How Xt Xs done

# Undo and Redo

Undo in vi is done by `u`

- Or to do in last line mode you could type in `:undo`
- `U` undo all latest changes on one line

Redo in vi is done by `CTRL` `R`

- Or to do in last line mode you could type in `:redo`

# Simple Tutorial: From Start to quit

This simple tutorial illustrates how to write, delete, copy, paste, replace, save, and quit. Start `vi` by `vi newfile.txt` and type the following

i This is how we write esc o and copy lines esc k y y j
p k y 3 w j ) a space bar esc p o ummm esc b
x r E l r N l r D : w q enter

This will produce following and goes back to command prompt

```
and copy lines
This is how we write and copy lines
END
```

# Simple Tutorial: From Start to quit

i This is how we write esc o and copy lines esc k y y j
p k y 3 w j ) a space bar esc p o ummm esc b
x r E l r N l r D : w q enter

The command in the tutorial

i insert esc back to command mode o add new line after current line k move cursor up yy copy a line j move cursor down p paste after cursor point y3w copy three words ) move to end of sentence a append b move cursor to previous word x erase a character r replace a character l move cursor right :wq write to a file and quit

# Learn by experience

```
1  The five boxing wizards jump quickly.
2  See the quick brown fox jump over the lazy dog.
3  A mad boxer shot a quick, gloved jab to the jaw of his dizzy opponent.
4  We promptly judged antique ivory buckles for the next prize.
5  A quart jar of oil mixed with zinc oxide makes a very bright paint.
6  The job requires extra pluck and zeal from every young wage earner.
```

Complete all tasks with minimum number of retyping, but with commands

- Substitute all j's to z and all z's to j
- Copy lines 1, 3, 5, and 6, and make new paragraph with those lines
- Delete three words "requires extra pluck," and type in "need lot of money" in the place
- Add "caps" at the end of all words with "w", e.g., wizards to "wizardscaps"

# Vi Configurations

Place `.vimrc` to your home direcotry

Have a look at the sample file
https:

//github.com/resourceful/lecture_sysprog/blob/master/01-intro/misc/.vimrc

# References

Graphical cheat sheet of Vi and VIM

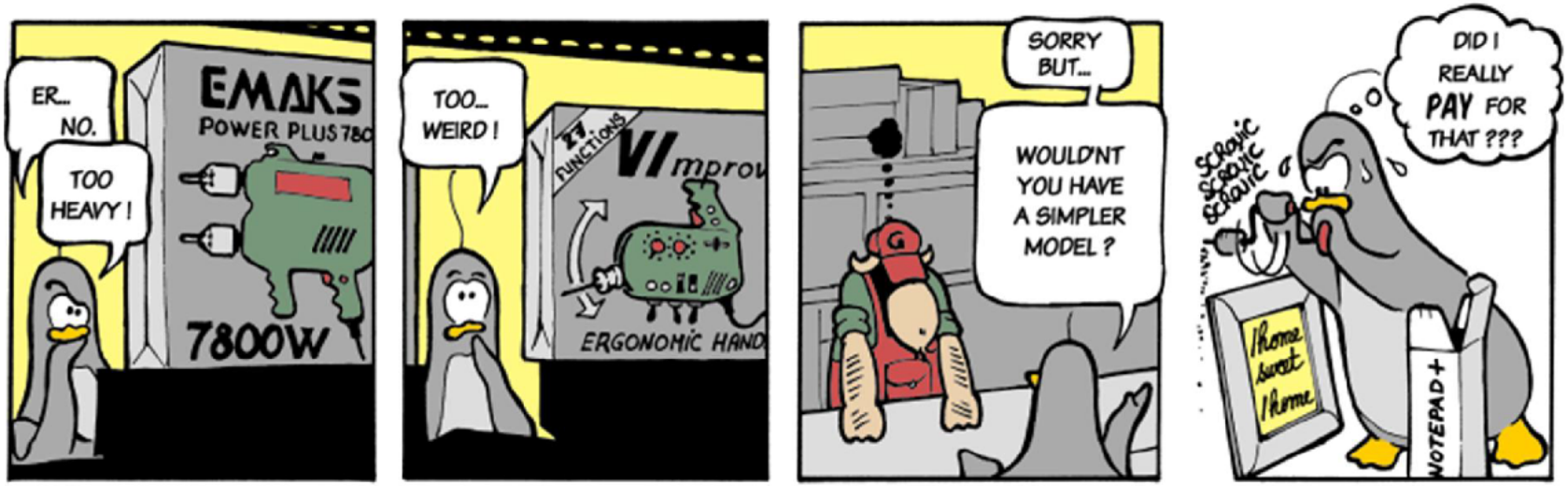
- http://www.viemu.com/a_vi_vim_graphical_cheat_sheet_tutorial.html

Cursor movement Commands

- http://www.kcomputing.com/vi.html

List of Commands

- http://www.smashingmagazine.com/2010/05/03/vi-editor-linux-terminal-cheat-sheet-pdf/




He knows so many commands!
Such dextrous fingers!
I'd let him macro me any time!

# What is Emacs

Emacs (Editor MACroS) is the extensible, customizable, self-documenting, real-time display editor

Richard Stallman is the author of Emacs; the author of GCC and GDB

Runs on LISP engines + lots of LISP libraries

**Richard Stallman**
**The founder of GNU**

http://www.theregister.co.uk/

# What is Emacs and why use it? (cont'd)

It is not the only good choice, there are options like VI, VIM Works on many platforms and independent of GUI

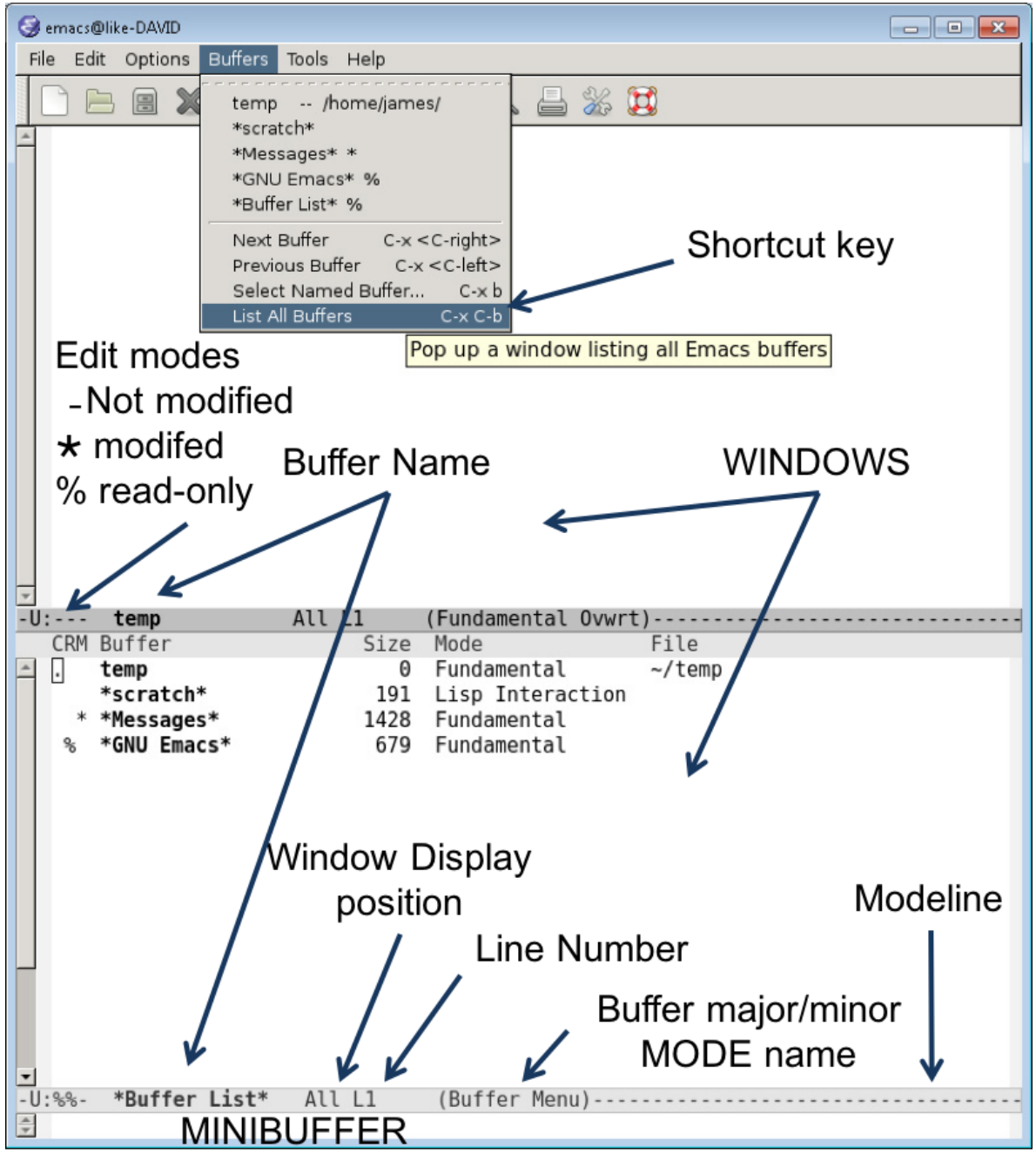
Extremely powerful

vi often does things with fewer keystrokes, but emacs easily surpass vi when it comes to searching and replacing and using macros

# What is Emacs and why use it? (cont'd)

Some of assumptions of Emacs are

- No mouse! – Much more reliable and much faster for experienced user
- No particular keyboard; No particular GUI environment
- Runs through telnet (as well as directly)


emacs@like-DAVID
File Edit Options Buffers Tools Help
temp -- /home/james/
*scratch*
*Messages* *
*GNU Emacs* %
*Buffer List* %
Next Buffer C-x <C-right>
Previous Buffer C-x <C-left>
Select Named Buffer... C-x b
List All Buffers C-x C-b
Shortcut key
Pop up a window listing all Emacs buffers
Edit modes
-Not modified
* modifed
% read-only
Buffer Name
WINDOWS
-U:--- temp All L1 (Fundamental Ovwrt)
CRM Buffer Size Mode File
temp 0 Fundamental ~/temp
*scratch* 191 Lisp Interaction
* *Messages* 1428 Fundamental
% *GNU Emacs* 679 Fundamental
Window Display position
Line Number
Modeline
Buffer major/minor MODE name
-U:%%- *Buffer List* All L1 (Buffer Menu)
MINIBUFFER

# Emacs Preliminaries

In the emacs documentation, key sequences described as:

- C-e – This is Ctrl-e
- C-x C-b – This is Ctrl-x Ctrl-b
- ^b - this is Ctrl-b
- C-x b – This is Ctrl-x b
- M-e – This is Meta-e or Alt-e

On the PC, you can use the Alt key or Esc-release to substitute Meta key

When you press a valid key sequence, emacs executes a command associated with the key

# Moving Around

Emacs uses the control keys to move in the four directions

Ctrl-b ← Ctrl-n ↓ Ctrl-p ↑ Ctrl-f →

To move by units of word, sentence, and paragraph

Meta-b Previous word Meta-f Next word Meta-a Previous sentence Meta-e Next sentence Meta-{ Previous paragraph Meta-} Next paragraph

# Deleting

Delete a word, line, and sentence

Ctrl - d Delete a character Meta - d Delete a word Ctrl - k Delete a line
Meta - k Delete a sentence

## When in Doubt

Use "Get me out of here" command Ctrl - g

# Searching

- Ctrl - s asks for searh pattern
  - Ctrl - s to search next pattern
  - Ctrl - r to search previous pattern
  - Regular expressions can be also used in searching with Meta - s

# Substitution

Meta - % to replace or Meta -replace-string

- Requests for search pattern; press enter for substituting pattern
- Replacing the substituting pattern this once SPC
- Skipping to the next without replcacing DEL
- Replace all remaining matches !
- Exiting replace command by RET

# Undo and Redo

Undo an unwanted change is done by Ctrl-_ Redo is reverse of undo,

undo direction is reversed by Ctrl-g and Ctrl-_

# Macro

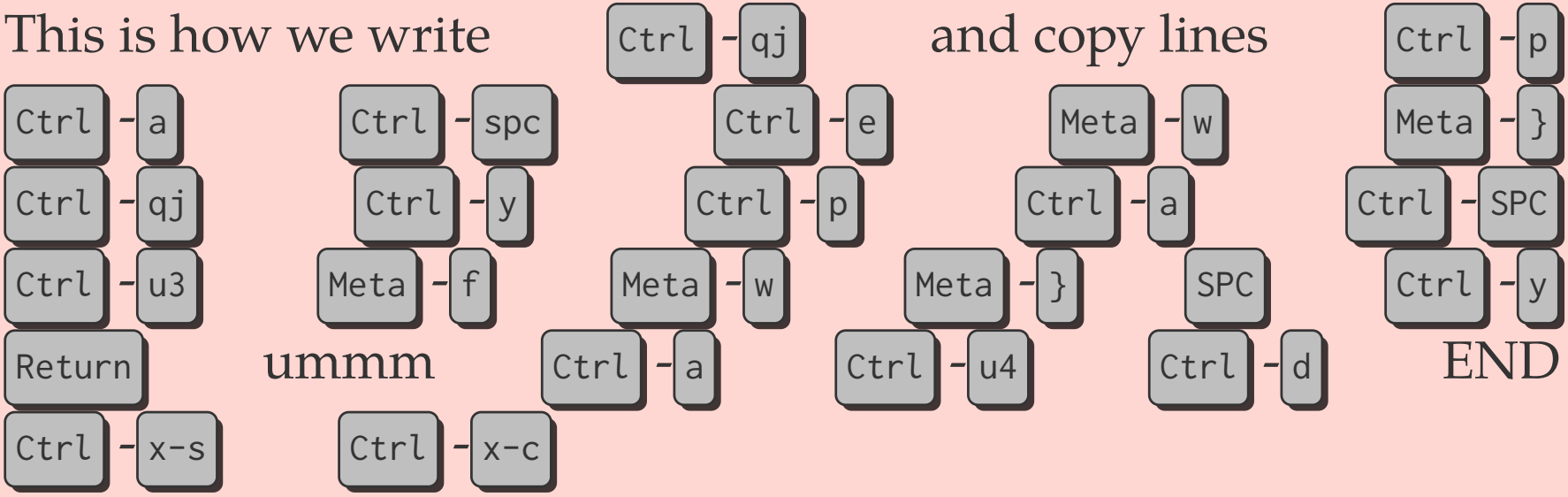
Macros are useful for repeatable key sequences that may be include commands.

Common macro commands

- Ctrl-x - ( – begin macro definition (after this, type whatever actions you would like repeated and stored)
- Ctrl-x - ) – end macro definition
- Ctrl-x - e – execute stored macro
- Ctrl-u5 Ctrl-e – execute stored macro 5 times (Note: Ctrl-u5 can prefix any emacs cmd, even a non-cmd)

# Simple Tutorial: From Start to quit

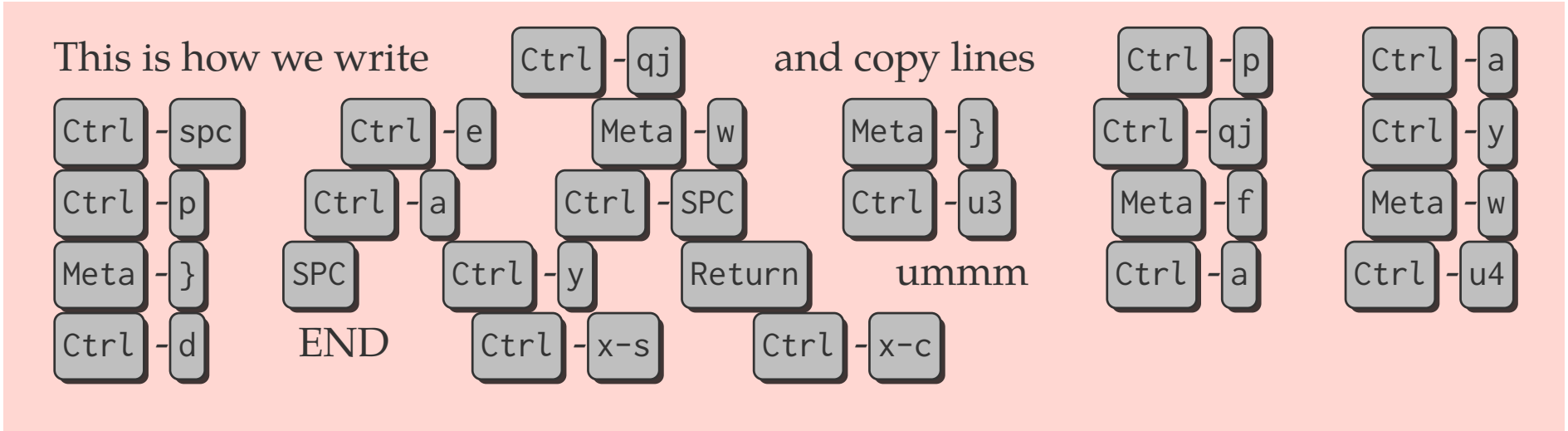
One can type without having to use complex commands but ....




This will produce following and goes back to command prompt

```
and copy lines
This is how we write and copy lines
END
```

# Simple Tutorial: From Start to quit




## Explaining the commands in the tutorial

Ctrl-qj Add a new line Ctrl-p Move to prvious line Ctrl-a Move to beginning of the sentence Ctrl-spc Start highlighting Ctrl-e Move to end of sentence Meta-w Copy highlighted Meta-} Move to end of the paragraph Ctrl-y Paste copied text Ctrl-u3 Repeat command with following number Meta-f Move to next word Ctrl-d Delete a character Ctrl-x-s Save the document Ctrl-x-c Quit

# Learn by experience

```
1  The five boxing wizards jump quickly.
2  See the quick brown fox jump over the lazy dog.
3  A mad boxer shot a quick, gloved jab to the jaw of his dizzy opponent.
4  We promptly judged antique ivory buckles for the next prize.
5  A quart jar of oil mixed with zinc oxide makes a very bright paint.
6  The job requires extra pluck and zeal from every young wage earner.
```

Complete all tasks with minimum number of retyping, but with commands

- Substitute all j's to z and all z's to j
- Copy lines 1, 3, 5, and 6, and make new paragraph with those lines
- Delete three words "requires extra pluck," and type in "need lot of money" in the place
- Add "caps" at the end of all words with "w", e.g., wizards to "wizardscaps"

# emacs Configurations

Place `.emacs` to your home direcotry

Have a look at the sample file
https:

//github.com/resourceful/lecture_sysprog/blob/master/01-intro/misc/.emacs

# References

Reference card with most commands you'll ever need

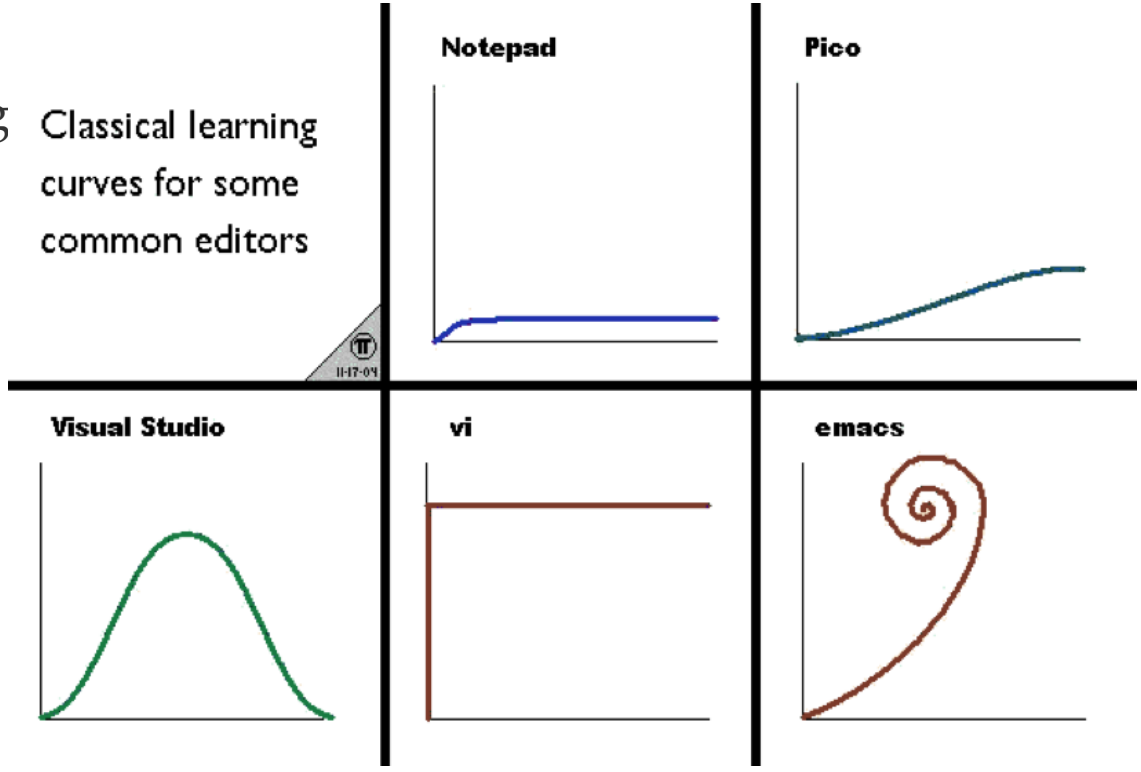
- http://home.uchicago.edu/~gan/file/emacs.pdf

Official GNU emacs site

- http://www.gnu.org/software/emacs/

An emacs HowTo

- https://www.emacswiki.org




# Homework

- Read up to chapter 3 File I/O — There will be a pop quiz on chapter 3
- Download source code for F2FS file system in advance, we will going to learn how to use ctag