HỆ ĐIỀU HÀNH LINUX

Đỗ Thanh Nghị
dtnghi@cit.ctu.edu.vn

Tổng quan

- Lịch sử
  - UNIX
  - LINUX
- Đặc điểm tổng quát
- Một số ứng dụng

Lịch sử

- UNIX
  - Được thiết kế và cài đặt từ những năm 1960 (tác giả: Ken Thompson) tại Bell Labs (AT&T) dành cho minicomputers và mainframes) và phiên bản đầu tiên công bố vào năm 1970
  - Một trong những hệ điều hành phổ biến nhất vì tính đơn giản và dễ tương thích
  - Là nguồn cảm hứng cho các hệ điều hành sau này

Lịch sử

- Đôi dòng lịch sử
  - 1973, được viết lại bằng ngôn ngữ C (do Dennis Ritchie phát triển)
  - 1975, phân phối phiên bản V6 cho các trường đại học nổi tiếng nhất (trong đó có Berkeley)
  - 1979, phân phối phiên bản V7 rộng rãi trong lĩnh vực công nghiệp ➔ vấn đề tương thích và các phiên bản **tựa UNIX** (**UNIX-like**) ra đời
  - Những năm 1980: thời kỳ công nghiệp của UNIX
    - Hệ thống V của AT&T cho phép thương mại hóa
    - 4.2 BSD của Berkely: là hệ thống cơ sở cho nhiều công ty trong đó có Sun Microsystems (SunOS), Digital (Ultrix), …
      - Berkeley phát triển riêng một phiên bản khác có tên BSD (Berkeley Software Distribution) với kỹ thuật phân trang bộ nhớ, dịch vụ mạng (TCP/IP) và các thành phần bổ sung khác.
    - XENIX của Microsoft cho các microcomputers (tương tích với hệ thống V)

Lịch sử

- Các sự kiện
  - Sự ra đời của UNIX
  - Năm 1983, Richard Stallman bắt đầu dự án GNU project với mục đích tạo ra một hệ điều hành tự UNIX
  - Giấy phép GPL (GNU General Public License)
  - Những năm đầu thập kỷ 1990, dự án GNU đã có hầu như đủ các phần mềm cần thiết để tạo nên một hệ điều hành hoàn chỉnh. Tuy nhiên nhân của hệ điều hành GNU (Kernel Hurd) chưa hoàn chỉnh không hấp dẫn được các nhà phát triển ➔ HĐH GNU đến giờ vẫn chưa hoàn thành.

Lịch sử

- Các sự kiện
  - Một dự án HĐH tự do khác được phát triển vào những năm 1980 tại **University of California, Berkeley** (phiên bản 6 của UNIX) với tến gọi BSD. Tuy nhiên BSD sử dụng mã nguồn của UNIX nên phải tuân theo luật của AT&T. Điều này hạn chế sự phát triển của BSD
  - MINIX, một HĐH tựa UNIX do Andrew S. Tanenbaum phát triển năm 1987, dự định dành cho môi trường học tập và nghiên cứu. Trong khi mã nguồn được để mở, thi việc thay đổi và phân phối lại mã nguồn của HĐH bị hạn chế. Thêm vào đó, MINIX được thiết kế cho kiến trúc 16 bits, không tương thích với các kiến trúc 32 bits.
  - Các lý do trên đã thúc đẩy Linus Torvalds bắt đầu dự án của mình.

Lịch sử

- Sự ra đời của Linux
  - Năm 1991, tại Helsinki, Phần Lan, Linus Torvalds bắt đầu một dự án (sau này trở thành nhân Linux – Linux kernel).
  - Bắt đầu từ một thiết bị đầu cuối (terminal emulator) mà Torvalds sử dụng để truy cập các servers UNIX của trường Đại học.
  - Torvalds viết các chương trình đặc biệt dành cho phần cứng độc lập với HĐH vì anh muốn sử dụng các chức năng của máy tính cá nhân (PC) mới của mình (với bộ vi xử lý 80386)
  - Chương trình này được viết trên MINIX bằng ngôn ngữ C và dịch bằng GNU C

Linus Torvalds

Lịch sử

- Sự ra đời của Linux
  - Ngày 25/8/1991, Torvalds thông báo hệ thống này trên hệ thống thông báo Usenet với tựa "comp.os.minix.":

    Hello everybody out there using minix -

    I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

    I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

    Linus (torvalds@kruuna.helsinki.fi)

    PS. Yes – it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

    —Linus Torvalds

Lịch sử

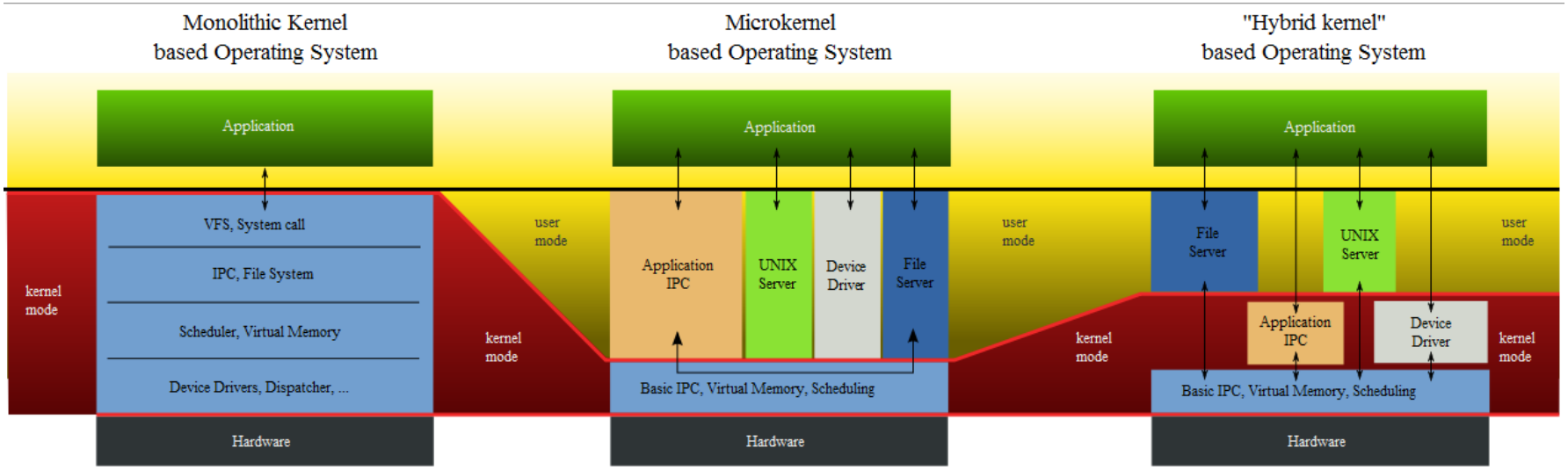
- Tên gọi
  - Linus Torvalds muốn gọi tác phẩm của mình là Freax, kết hợp của “freak” (kỳ dị), "free” (tự do), and “x” (Unix). Trong quá trình làm việc anh lưu chương trình với tên “Freax” khoảng nửa năm.
  - Để việc phát triển dễ dàng, các tập tin được upload lên một FTP server (ftp.funet.fi) của FUNET vào tháng 9 năm 1991. Ari Lemmke, cộng sự của Torvald tại ĐH Helsinki, nghĩ rằng tên Freax không hay lắm nên đặt lại thành “Linux” mà không hỏi ý Torvalds. Sau đó Torvalds đồng ý với tên gọi mới này.

Đặc điểm tổng quát

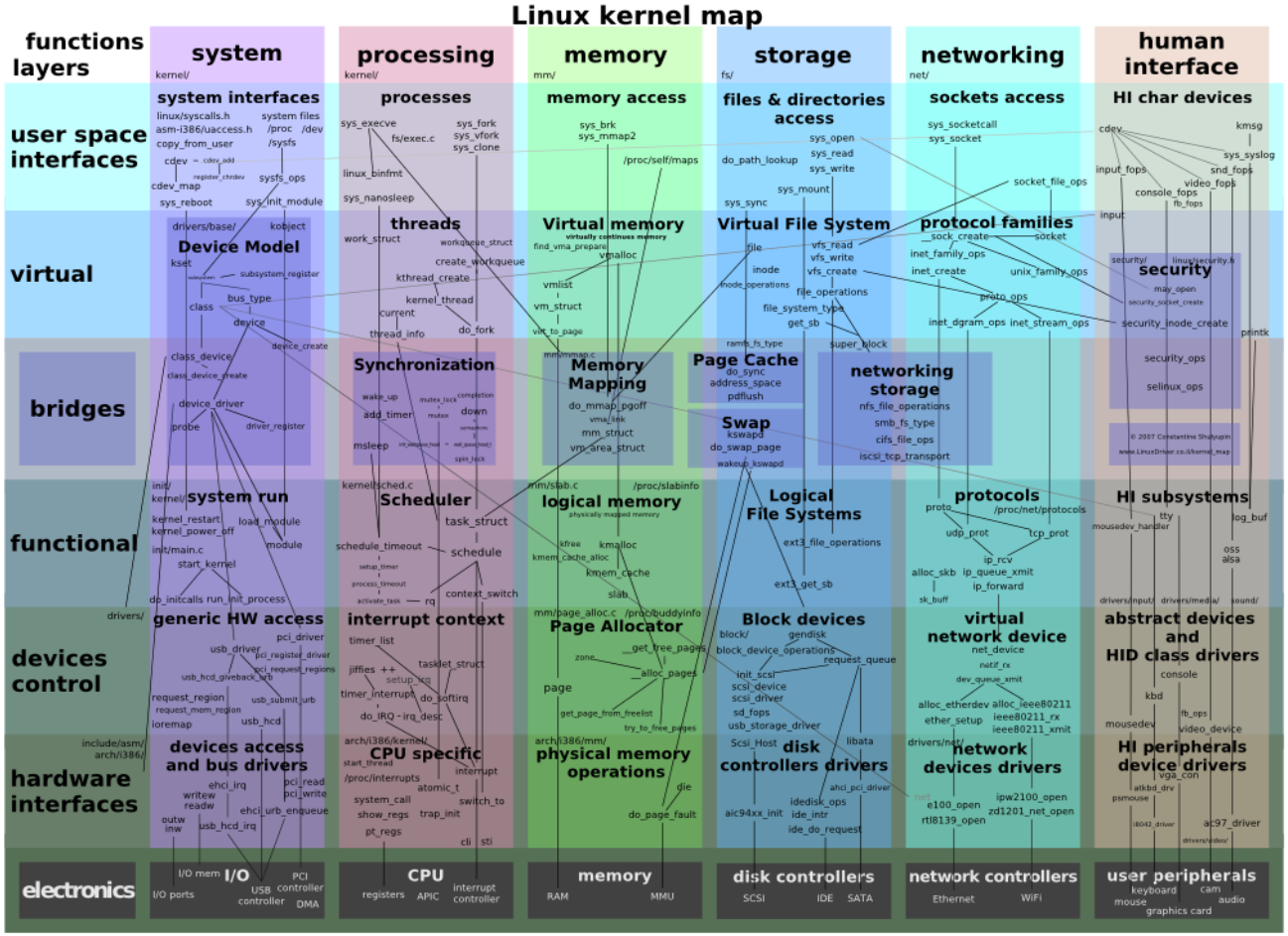
- Hệ thống Linux
  - Nhân
    - Monolithic
    - Các modun khả nạp
  - Các ứng dụng và tiện ích: chủ yếu từ dự án GNU
  - Tên gọi đúng phải là: GNU/Linux
  - Các distro: RedHat, Fedora, Suse, Slackware, Knoppix, Mandriva, Ubuntu, … (xem thêm tại: http://www.gnu.org/distros/free-distros.html)
- Giấy phép
  - Nhân và đa số các ứng dụng được phân phối với giấy phép GPL của GNU
    - Phân phối đến người sử dụng cùng với mã nguồn
    - Mã nguồn có thể được sửa đổi cho mục đích của công việc
    - Tất cả lập trình viên trên toàn cầu có thể tham gia phát triển
    - Không trả tiền cho bản quyền
  - Ngoài ra, có thể có một số ứng dụng được phân phối với giấy phép khác

Đặc điểm tổng quát

- Các loại nhân



Đặc điểm tổng quát



Đặc điểm tổng quát

- Viết bằng ngôn ngữ C
- Chạy trên nhiều nền khác nhau: Alpha, AMD, Intel, MIPS, PowerPC, Sparc, …
- Kích thước tối đa bộ nhớ: 12 TB
- Kích thước tối đa hệ thống file: 50 TB (ext4)
- Kích thước tối đa file: 16 TB (ext4)
- Chạy trên hệ thống tối đa: 288 processors
- Đa quá trình
- Đa người dùng
- Hệ thống an toàn, ổn định, rất ít virus
- Nếu có lỗi, cộng đồng sẽ chữa lỗi
- Chứng chỉ LPI

Ứng dụng trên Linux

- Văn phòng (open office)
- Giải trí (movie player, xmms, totem player kaffeine, …)
- Xử lý ảnh (GIMP)
- Dịch vụ mạng (Telnet, SSH, FTP, Postfix, Apache, Bind, CUPS, OpenLDAP, Iptable, Squid, Mozilla-Firefox, SAMBA, NFS)
- Cơ sở dữ liệu (MySQL, PostgreSQL)
- Lập trình (Emacs, C/C++, QT Trolltech, Fortran, Java, R, octave, Lapack, Blas, Python, Perl, AWK, TCL/TK, PHP, …)
- Quản trị hệ thống(Webmin, VNC, …), …

LibreOffice

- LibreOffice
  - Là 1 bộ phần mềm văn phòng đa ngôn ngữ, đa nền và là phần mềm nguồn mở.
  - Tương thích với hầu hết các phần mềm văn phòng khác (ví dụ: Ms Office)
  - Hỗ trợ unicode
  - Download, sử dụng và phân phối miễn phí
  - Web site: http://www.libreoffice.org/
  - Phiên bản mới nhất: > LibreOffice 5.0.3

LibreOffice

16

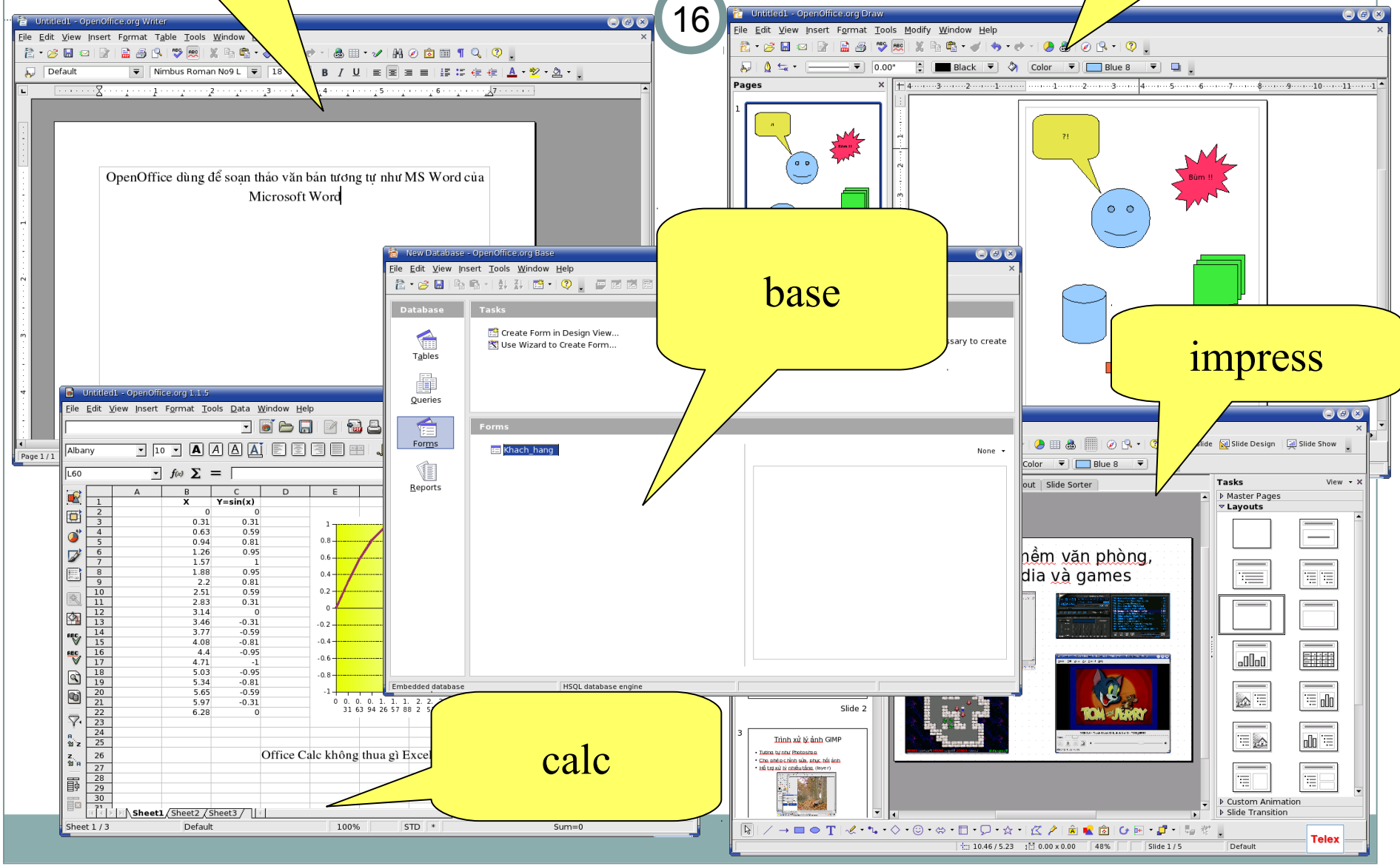
writer

draw

base

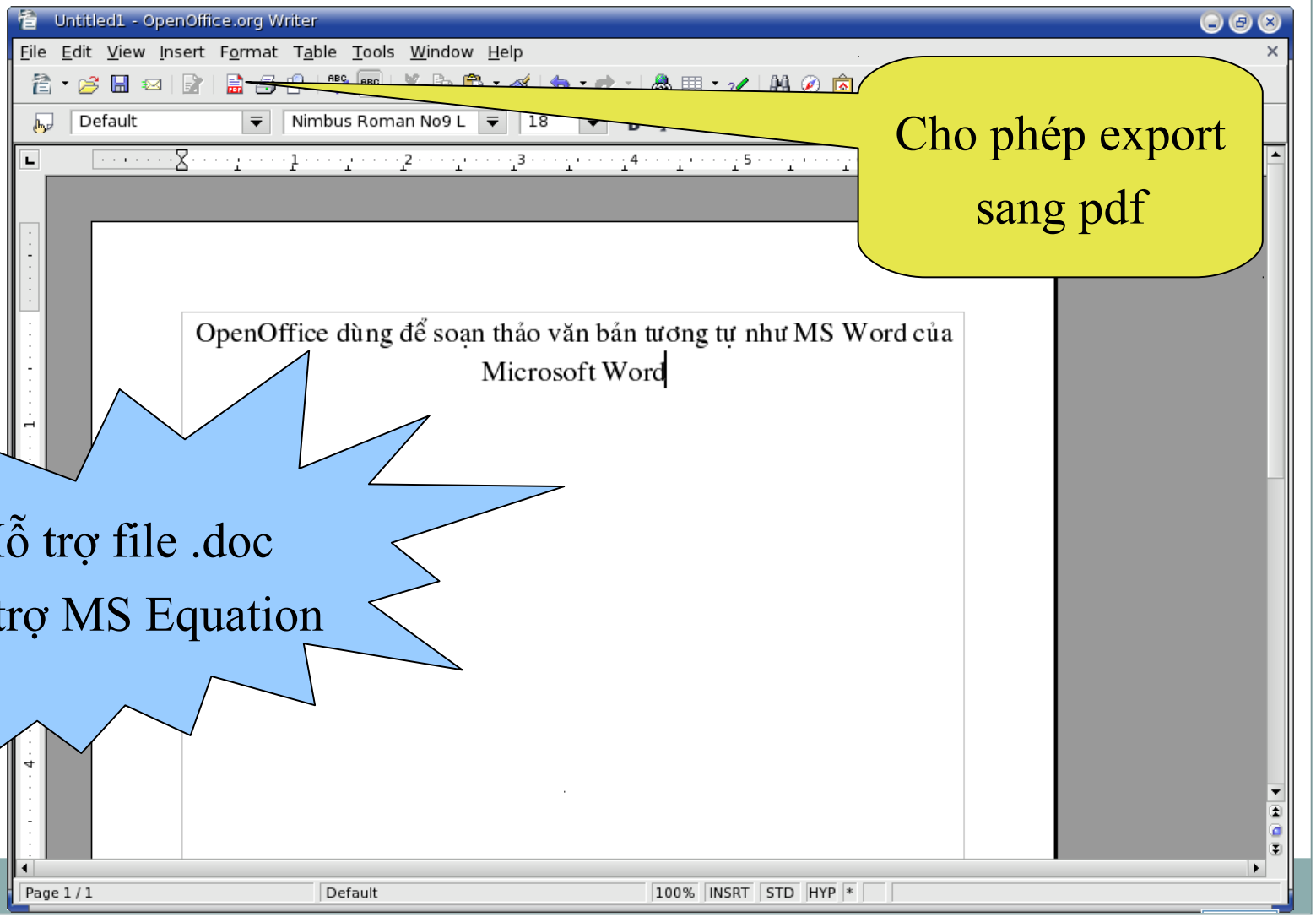
impress

calc

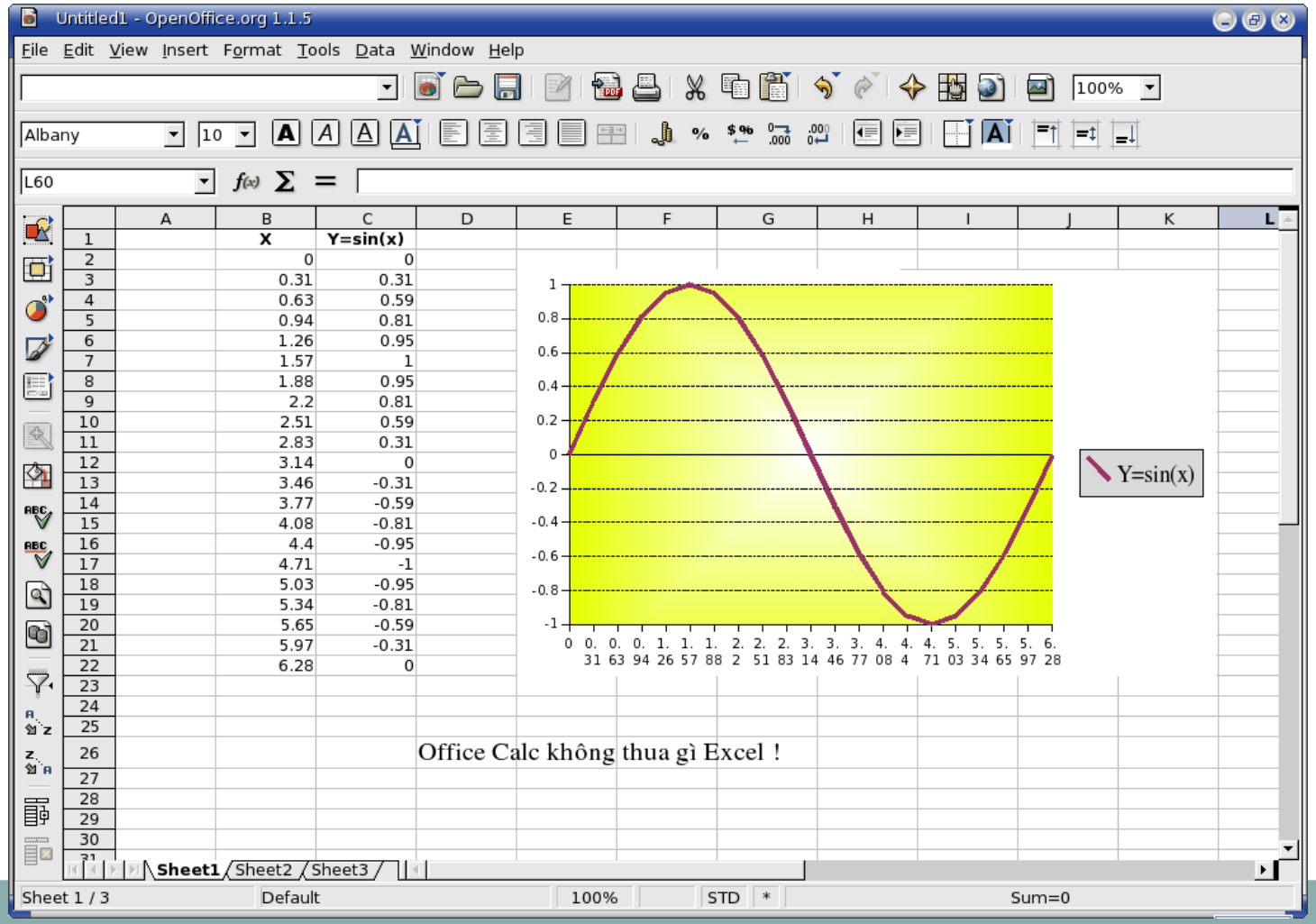


Writer (~Ms Word)

17

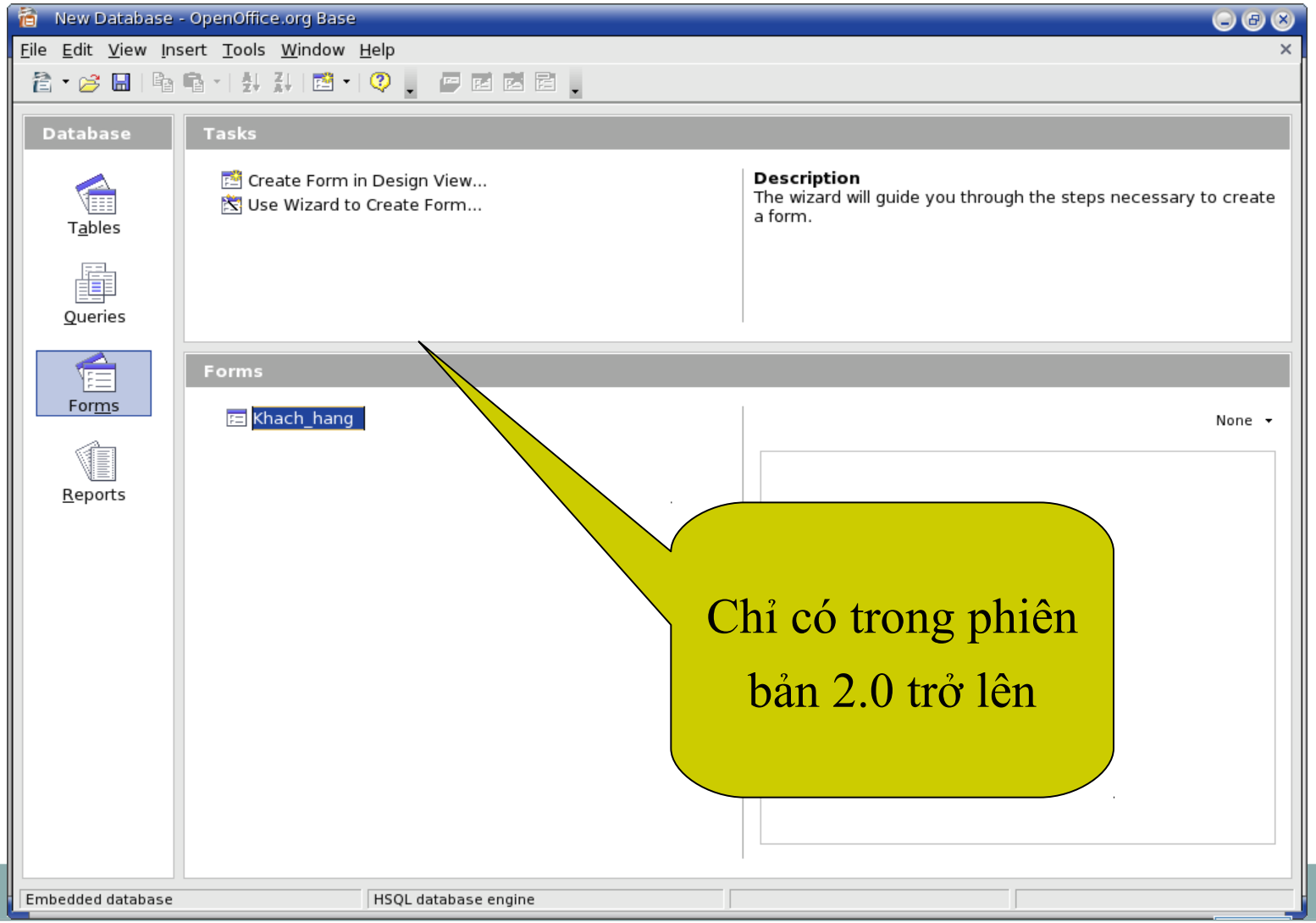


Calc (~Ms Excel)

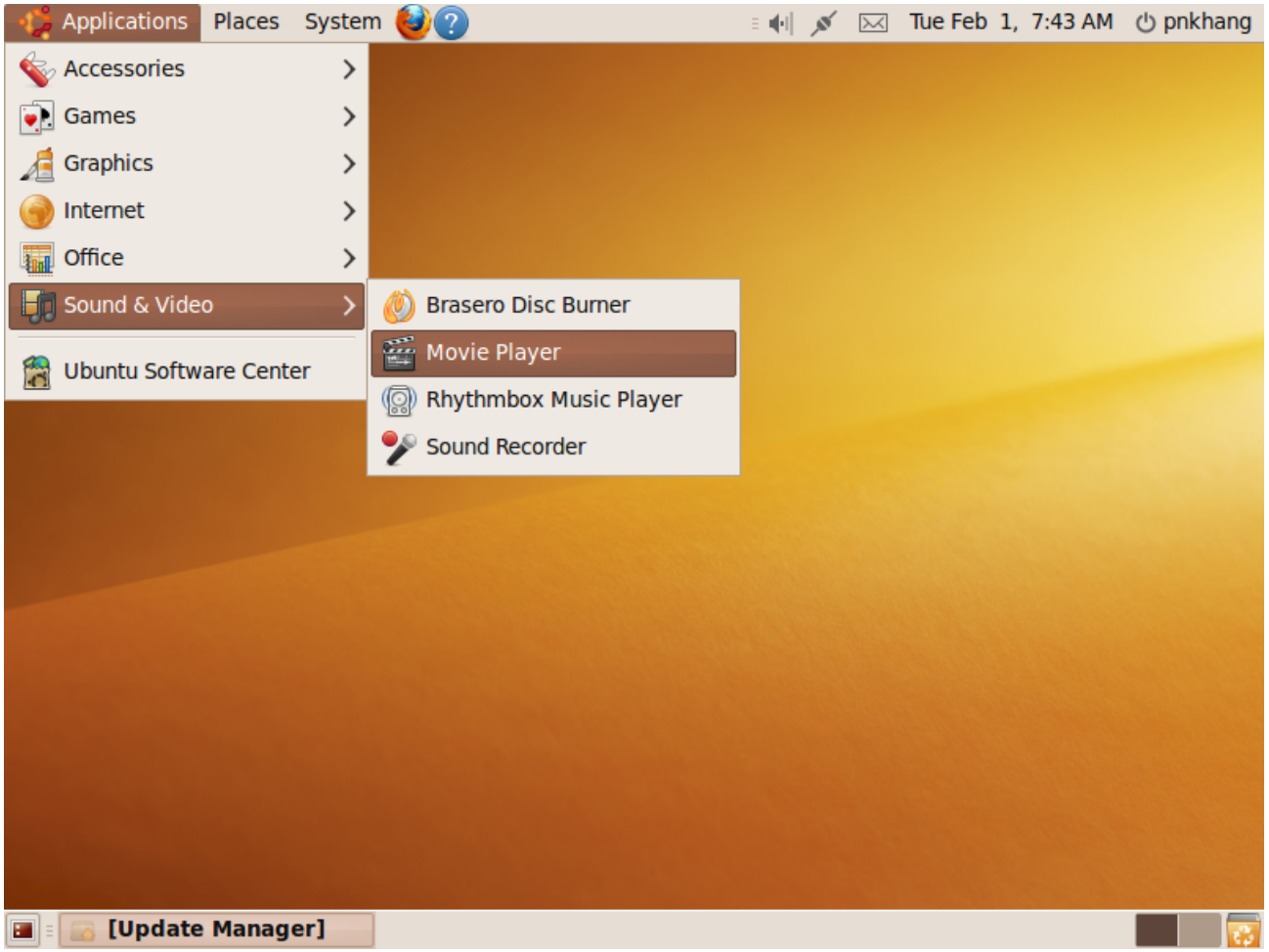


	X	Y=sin(x)
2	0	0
3	0.31	0.31
4	0.63	0.59
5	0.94	0.81
6	1.26	0.95
7	1.57	1
8	1.88	0.95
9	2.2	0.81
10	2.51	0.59
11	2.83	0.31
12	3.14	0
13	3.46	-0.31
14	3.77	-0.59
15	4.08	-0.81
16	4.4	-0.95
17	4.71	-1
18	5.03	-0.95
19	5.34	-0.81
20	5.65	-0.59
21	5.97	-0.31
22	6.28	0

Base (~Ms Access)



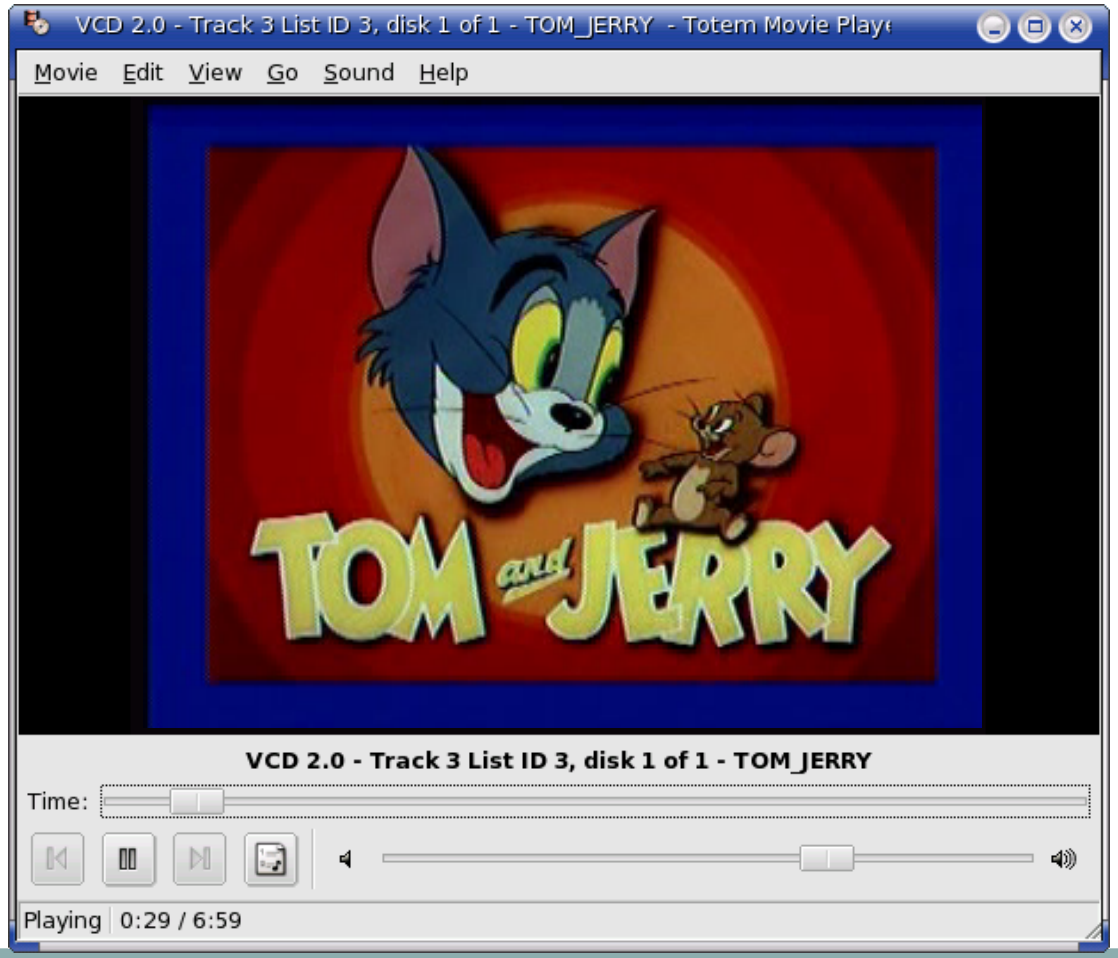
Movie player



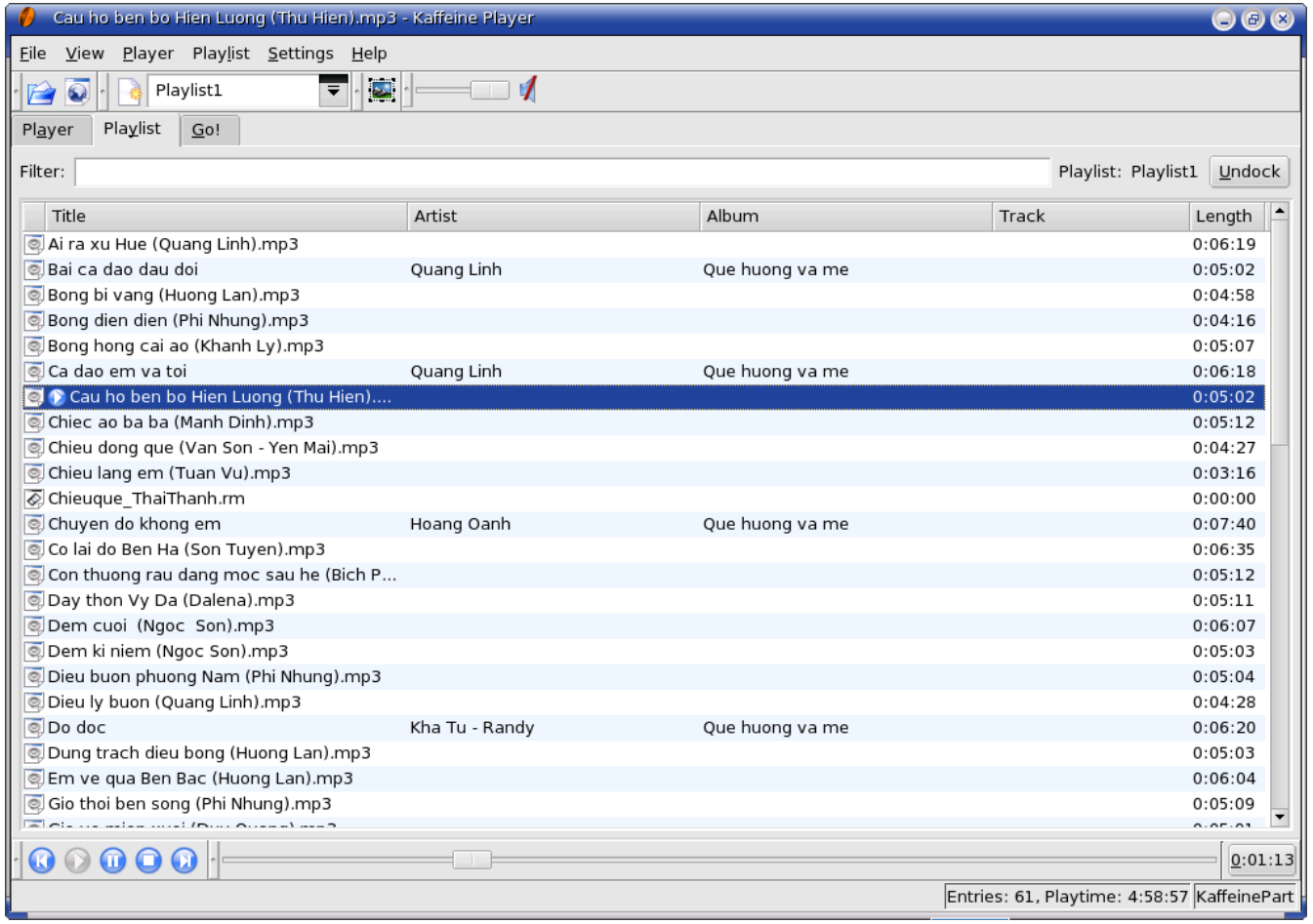
XMMS Player


X MULTIMEDIA SYSTEM
01:03
g Linh - Ca dao em va toi (6:18)
128 kbps
44 kHz
MONO
STEREO
RAND.
REP
X MULTIMEDIA SYSTEM - EQUALIZER
ON
AUTO
PRESETS
+20db
+0db
-20db
PREAMP
60
170
310
600
1K
3K
6K
12K
14K
16K
PLAYLIST EDITOR
134. Ai ra xu Hue (Quang Linh) 6:18
135. Quang Linh - Bai ca dao dau doi 5:01
136. Bong bi vang (Huong Lan) 4:58
137. Bong dien dien (Phi Nhung) 4:16
138. Bong hong cai ao (Khanh Ly) 5:06
139. Quang Linh - Ca dao em va toi 6:18
140. Cau ho ben bo Hien Luong (Thu Hien) 5:02
141. Chiec ao ba ba (Manh Dinh) 5:12
142. Chieu dong que (Van Son - Yen Mai) 4:26
143. Chieu lang em (Tuan Vu) 3:16
144. Hoang Oanh - Chuyen do khong em 7:40
145. Co lai do Ben Ha (Son Tuyen) 6:34
146. Con thuong rau dang moc sau he (Bich Ph... 5:12
+ FILE
- FILE
SEL ALL
MISC OPT.
6:18/11:42:13+
01:03
LOAD LIST

Totem Player



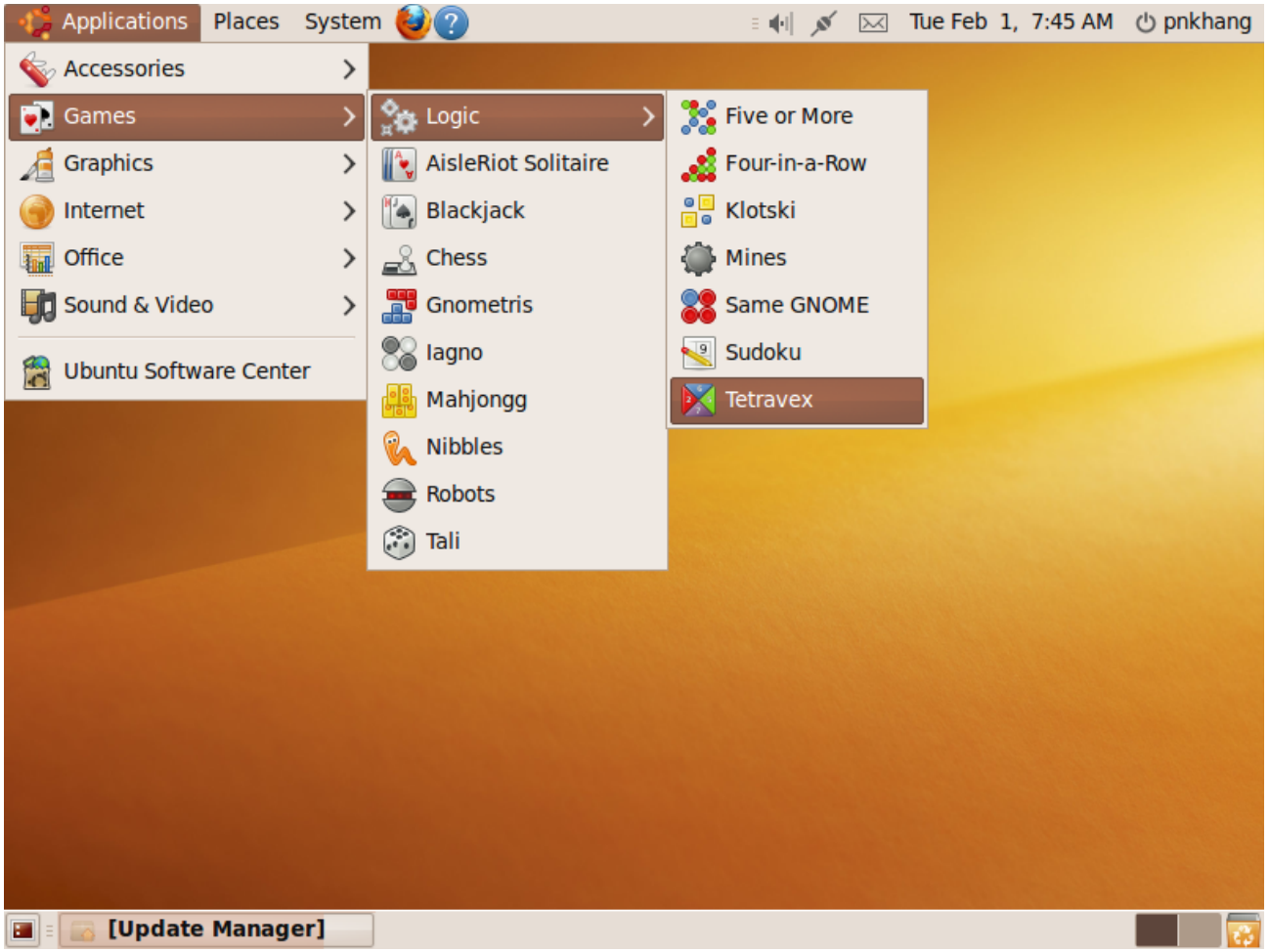
Kaffeine


Cau ho ben bo Hien Luong (Thu Hien).mp3 - Kaffeine Player
File View Player Playlist Settings Help
Playlist1
Player Playlist Go!
Filter:
Playlist: Playlist1
Undock
Title Artist Album Track Length
Ai ra xu Hue (Quang Linh).mp3 0:06:19
Bai ca dao dau doi Quang Linh Que huong va me 0:05:02
Bong bi vang (Huong Lan).mp3 0:04:58
Bong dien dien (Phi Nhung).mp3 0:04:16
Bong hong cai ao (Khanh Ly).mp3 0:05:07
Ca dao em va toi Quang Linh Que huong va me 0:06:18
Cau ho ben bo Hien Luong (Thu Hien).... 0:05:02
Chiec ao ba ba (Manh Dinh).mp3 0:05:12
Chieu dong que (Van Son - Yen Mai).mp3 0:04:27
Chieu lang em (Tuan Vu).mp3 0:03:16
Chieuque_ThaiThanh.rm 0:00:00
Chuyen do khong em Hoang Oanh Que huong va me 0:07:40
Co lai do Ben Ha (Son Tuyen).mp3 0:06:35
Con thuong rau dang moc sau he (Bich P... 0:05:12
Day thon Vy Da (Dalena).mp3 0:05:11
Dem cuoi (Ngoc Son).mp3 0:06:07
Dem ki niem (Ngoc Son).mp3 0:05:03
Dieu buon phuong Nam (Phi Nhung).mp3 0:05:04
Dieu ly buon (Quang Linh).mp3 0:04:28
Do doc Kha Tu - Randy Que huong va me 0:06:20
Dung trach dieu bong (Huong Lan).mp3 0:05:03
Em ve qua Ben Bac (Huong Lan).mp3 0:06:04
Gio thoi ben song (Phi Nhung).mp3 0:05:09
0:01:13
Entries: 61, Playtime: 4:58:57
KaffeinePart

Kaffeine

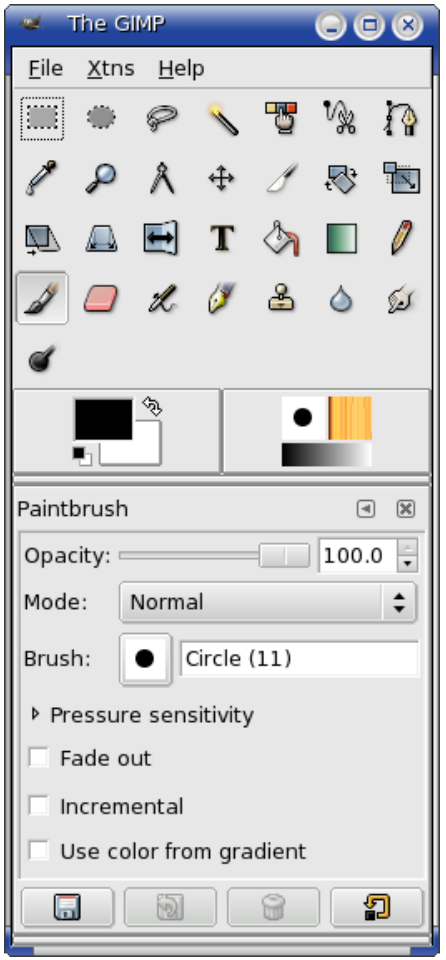

VCD 2.0 - Track 1 List ID 1, disk 1 of 1 - TOM_JERRY - Kaffeine Player
File View Player Playlist Settings Help
VCD
Player Playlist Go!
A Metro-Goldwyn-Mayer
CARTOON
COLOR BY TECHNICOLOR
0:00:04
Entries: 1, Playtime: 0:00:00
KaffeinePart

Game


Applications
Places
System
Tue Feb 1, 7:45 AM
pnkhang
Accessories
Games
Graphics
Internet
Office
Sound & Video
Ubuntu Software Center
Logic
AisleRiot Solitaire
Blackjack
Chess
Gnometris
Iagno
Mahjongg
Nibbles
Robots
Tali
Five or More
Four-in-a-Row
Klotski
Mines
Same GNOME
Sudoku
Tetravex
[Update Manager]

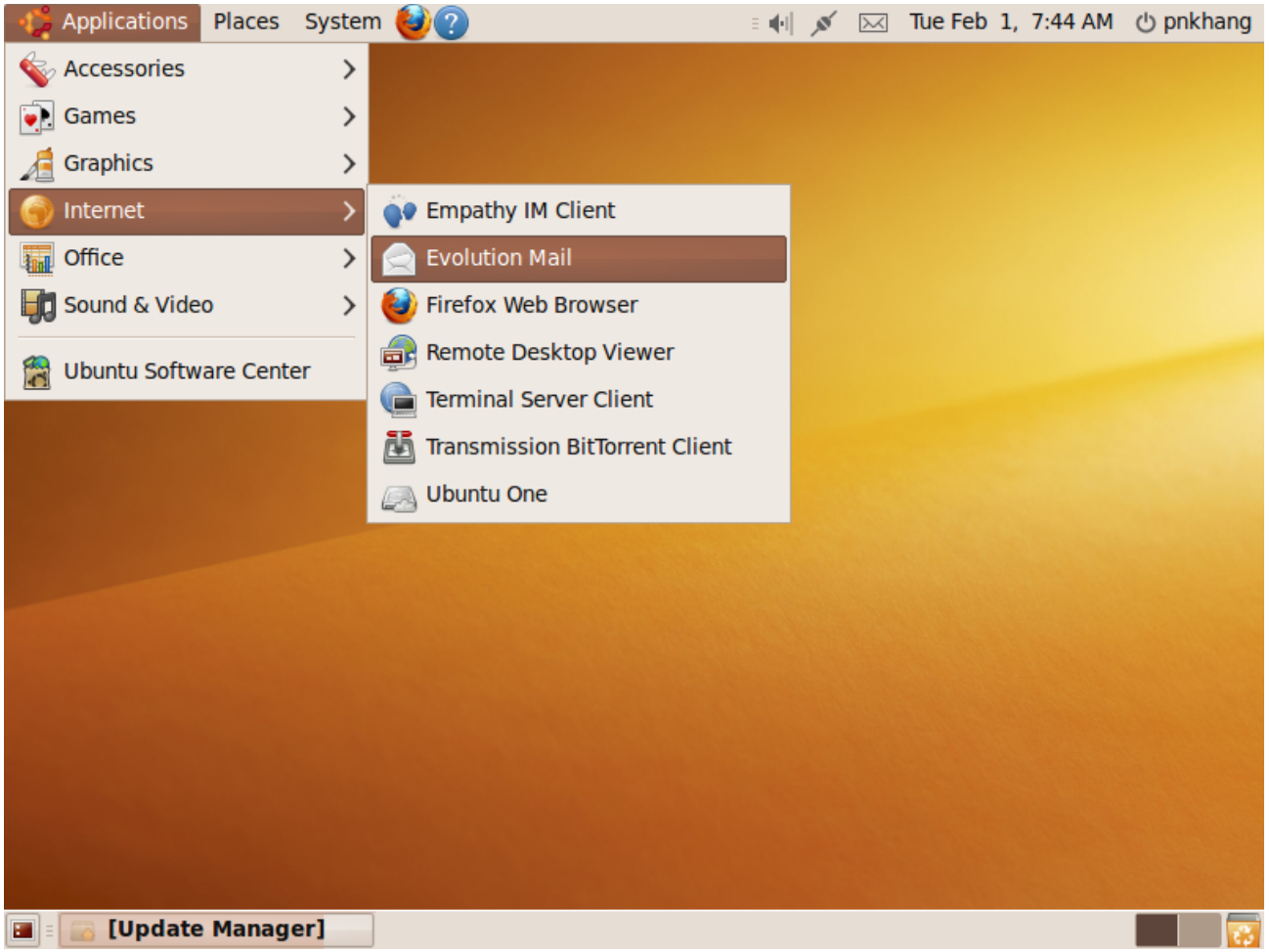
Trình xử lý ảnh The GIMP

- Tương tự như Photoshop
- Cho phép chỉnh sửa, phục hồi ảnh
- Hỗ trợ xử lý nhiều tầng (layer)

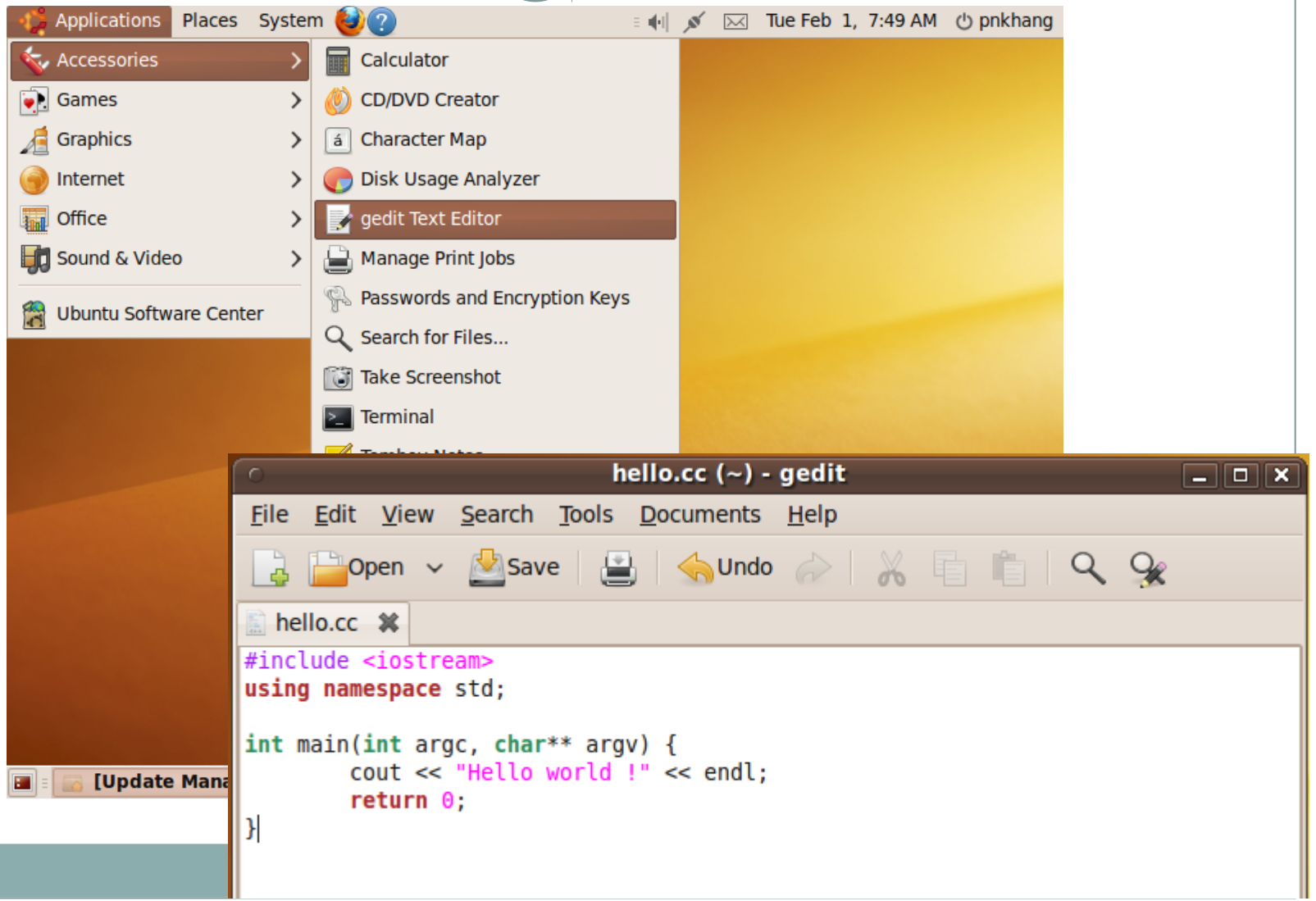


Trình duyệt web: Mozilla Firefox

Mail: Evolution mail



Trình soạn thảo: gedit



Đường dẫn

- Đường dẫn
  - Đường dẫn tuyệt đối: truy cập thư mục hay tập tin qua đường dẫn đầy đủ (bắt đầu với **/**), độc lập với vị trí thư mục hiện hành
  - Đường dẫn tương đối: truy cập thư mục hay file qua đường dẫn (không bắt đầu bằng **/**), phụ thuộc vào vị trí thư mục hiện hành
- Thư mục đặc biệt:
  - Thư mục gốc: **/**
  - Thư mục cha: **..**
  - Thư mục hiện hành: **.**
  - Thư mục cá nhân của người dùng: **~**

Lệnh cơ bản

- ls <thư mục>: liệt kê thư mục
  - ví dụ: ls /etc
- mkdir <thư mục>: tạo thư mục
  - ví dụ: mkdir toto
- cd <thư mục>: chuyển đổi thư mục
  - ví dụ: cd toto
- cp <nguồn> <đích>: sao chép
  - ví dụ: cp /etc/passwd .

Lệnh cơ bản

- mv <nguồn> <đích>: chuyển hay đổi tên file, thư mục
  - ví dụ: mv ./passwd toto/passwd.tmp
- rm <file/thư mục>: xóa file, thư mục
  - ví dụ 1: rm passwd
  - ví dụ 2: rm -R toto
- chmod <quyền> <file/thư mục>: đặt thuộc tính cho file, thư mục
  - ví dụ: chmod o+w toto/passwd.tmp

Lệnh cơ bản

- chown <sở hữu> <file/thư mục>: thay đổi chủ sở hữu file hay thư mục
  - ví dụ: chown nghi.profs toto
- cat , more <path/file>: xem tập tin text
  - ví dụ 1: cat /etc/passwd
  - ví dụ 2: more /etc/passwd
- vi: soạn thảo văn bản (đọc hướng dẫn trong phần trình soạn thảo vi)

Lệnh cơ bản

- head, tail, wc, tar, gzip, fdisk, rpm, ifconfig, route, init, useradd, passwd, df, du, ln, top, mount, etc.
  - ví dụ 1: head -10 /etc/passwd
  - ví dụ 2: wc -l /etc/passwd
  - ví dụ 3: tar -cvf toto.tar toto
  - ví dụ 4: gzip toto.tar
  - ví dụ 5: passwd
  - ví dụ 6: df -h
  - ví dụ 6: du -k dir
  - ví dụ 8 : top

Lệnh cơ bản

- man <section> <lệnh>: xem trợ giúp của lệnh, section = 1-user cmd, 2-sys calls, 3-sub routines, 4-devices, 8-sys admin
  - ví dụ: man ls
- su <user>: chuyển user
  - ví dụ: su tutu
- reboot (init 6 hoặc Ctrl-Alt-Del): khởi động lại hệ thống
- poweroff: shutdown hệ thống và tắt máy
- shutdown: shutdown hệ thống
  - ví dụ: shutdown -h now

Bộ lọc

- Xem nội dung
- Sắp xếp
- Dịch, lựa chọn
- Tìm kiếm
- Cắt văn bản
- Khác
- sed
- awk

Xem nội dung

- cat hiển thị nội dung 1 file
- head -n hiển thị n dòng đầu tiên của 1 file
- tail -n hiển thị n dòng cuối cùng của 1 file
- more hiển thị cả file có phân trang
- less tương tự more, nhưng cho phép quay lui, tìm kiếm
- wc thống kê, đếm số ký tự, từ và hàng
  - -c đếm số ký tự
  - -w đếm số từ
  - -l đếm số hàng

Sắp xếp

- sort [OPTIONS] [tập tin]
  - Sắp xếp nội dung tập tin theo 1 thứ tự nào đó
  - Nếu không chỉ rõ tập tin nào, sort sẽ sắp xếp luồng nhập chuẩn (sử dụng ống dẫn)
- OPTIONS
  - -n trường đang quan tâm là số (mặc định là chuỗi)
  - -r xếp theo thứ tự giảm dần
  - -tx x là ký tự phân cách (mặc định là ký tự TAB)
  - +bd bd là trường bắt đầu (kể từ 0)
  - -kt kt là trường kết thúc
- Ví dụ:
  - sort auto
  - sort -t: -n +4 -5 auto
  - sort -t: +0 -1 +3n -4 auto

Dịch

- **tr** [OPTIONS] tập_hợp_1 [tập_hợp_2] <file
  - Chuyển từng ký tự trong tập hợp 1 thành ký tự tương ứng trong tập hợp 2
  - Ví dụ: tr [a-z] [A-Z] </etc/passwd
- OPTIONS:
  - -d xóa bỏ ký tự nếu nó nằm trong tập hợp 1

Trích chọn

- cut [OPTIONS] file
  - Trích chọn 1 số trường trong file file
  - OPTIONS
    - -dx x là ký tự phân cách, mặc định là TAB
    - -fds ds là sách các trường cách nhau bằng dấu phẩy, tính từ 1
  - Ví dụ:
    - cut -d: -f1,5 /etc/passwd
    - cut -d: -f1-3 /etc/passwd
- uniq
  - Xóa bỏ những dòng trùng nhau, chỉ giữ lại 1

Tìm kiếm

- Việc tìm kiếm một từ hay nhiều từ trong một văn bản có thể được thực hiện bằng các dùng lệnh **grep**, **fgrep** hay **egrep**.
- Các từ khóa được dùng trong việc tìm kiếm được kết hợp từ các ký tự. Sự kết hợp này được gọi là *biểu thức chính quy*.
- Biểu thức chính quy cũng được dùng trong các ứng dụng khác như **sed** và **vi**.
- grep biểu_thức file

Biểu thức chính quy cơ bản

Ký tự	Ý nghĩa
x (hay bất cứ ký tự nào khác)	Chứa 1 ký tự x
\<KEY	Từ bắt đầu bằng KEY
WORD\>	Từ kết thúc bằng WORD
^	Bắt đầu dòng
$	Kết thúc dòng
[Range]	Một đoạn ký tự ASCII
[^c]	Không chứa ký tự c
\[	Ký tự [
cat*	Chứa ca hoặc cat hoặc catt, …
.	Bất kỳ ký tự nào

Biểu thức chính quy mở rộng

Ký tự	Ý nghĩa
A1|A2|A3	Chứa A1 hoặc A2 hoặc A3
cat+	Chứa cat hoặc catt, hoặc cattt, …
cat?	Chứa ca hoặc cat, hoặc catt, …

Họ lệnh grep

- grep cơ bản:
  - Sử dụng các *biểu thức chính quy cơ bản* để tìm kiếm
- egrep:
  - Sử dụng các *biểu thức chính quy mở rộng* để tìm kiếm
- fgrep (fast grep):
  - Không hỗ trợ biểu thức chính quy.

Làm việc với lệnh grep

- Cú pháp:
  - grep [option] PATTERN FILE

grep	Các tùy chọn chính
-c	Đếm số dòng thỏa mãn mẫu PATTERN
-f	Mẫu tìm kiếm được lấy từ tập tin
-i	Không phân biệt chữ hoa chữ thường
-n	Hiển thị số thứ tự của dòng thỏa mãn mẫu PATTERN
-v	Hiển thị tất cả các dòng không thõa mãn mẫu
-w	Tìm chính xác mẫu

Grep

- Ví dụ:
  - grep -v “^$” /etc/passwd
  - Hiển thị tất cả các dòng không rỗng của tập tin /etc/passwd

egrep và fgrep

- Sử dụng tương tự như grep
- egrep sử dụng biểu thức chính quy mở rộng
- Ví dụ: egrep "linux|^image" FILE

  Tìm các dòng có chứa từ linux hoặc bắt đầu bằng image
- fgrep không hỗ trợ biểu thức chính quy
- Ví dụ: fgrep "cat*" FILE

  Tìm các dòng có chứa chuỗi cat*

Tách file

- split -n file
  - Tách file file thành nhiều file con, mỗi file con có n dòng
  - Tên file con được đặt tên từ xaa đến xaz

Các lệnh khác

- cmp, diff
  - So sánh 2 file
- paste: nối từng hàng của 2 file lại với nhau
- join: nối từng hàng của 2 file theo 1 trường nào đó
  - join -1 FIELD -2 FILED FILE1 FILE2
- tee:
  - Copy đầu ra của lệnh trước xuống file, và chuyển đầu ra thành đầu vào của lệnh sau

sed

- Trình soạn thảo luồng (**s**tream-oriented **ed**itor)
- Thông dịch lệnh và thực thi lệnh
- Kết quả in ra dòng xuất chuẩn (màn hình)
- Cú pháp
  - **sed** [options] 'lệnh' FILE
  - **sed** [options] **-f** script FILE
  - Trong đó:
    - Một lệnh có dạng: [địa chỉ][,địa chỉ][!]lệnh[tham số]
    - script là file chứa lệnh

- Ví dụ:
  - **sed 's/xx/yy/g'** FILE
    - thay thế tất cả các chuỗi xx thành yy
  - sed '/BSD/d' FILE
    - xóa tất cả các dòng có chứa từ BSD
  - sed -n '/^BEGIN/,/^END/p' FILE
    - In các dòng nằm giữa BEGIN và END
  - sed '/SAVE/!d' FILE
    - Xóa tất cả các dòng không chứa từ SAVE
  - sed '/^BEGIN/,/^END/s/xx/yy/g' FILE
    - Thay thế xx thành yy trong khoảng từ BEGIN đến END

sed

- Địa chỉ có thể là:
  - Số dòng, ví dụ: 3 (dòng số 3)
  - Mẫu: đặt trong cặp //, ví dụ: /BEGIN/
- **!** đặt sau địa chỉ có nghĩa là trừ phần địa chỉ ra
- Lệnh
  - s/mau/thaythe/g thay thế **mau** thành **thaythe**
    - Nếu không có g, chỉ thay thế một lần cho 1 dòng
    - Có thể sử dụng _ hoặc : để thay thế cho /
    - Ví dụ: **s_mau_thaythe_g** hoặc **s:mau:thaythe:g**
  - d xóa
  - p in (sử dụng với option -n)

sed

- sed '/^$/d' FILE
  - Xóa tất cả các dòng trống
- sed 's/.*/cp & &.copied/' FILE
  - Tạo danh sách các lệnh copy để copy các file
  - & = mẫu so khớp
  - Có thể sử dụng cặp \(\) để nhóm các mẫu, để tham chiếu đến các nhóm này ta dùng **số thứ tự nhóm**
  - Ví dụ: Nếu FILE chứa
    - abc.jpg
    - 123.png
  - sed 's/\(.*\)\.jpg/convert \1.jpg \1.gif/g' FILE

sed

- s/mau/thaythe/flags
  - Ngoài cờ g, lệnh s còn thể được sử dụng với
    - Số nguyên n (vd 2) lệnh s sẽ được thực hiện với mẫu thứ n
    - Số nguyên và g (vd: 2g) thực hiện lệnh s từ mẫu thứ 2 trở đi
    - p in kết quả sau khi thay thế (nên sử dụng
    - với option -n)
    - w tênfile ghi kết quả ra file tênfile
  - Có thể kết hợp nhiều cờ lại với nhau nếu có ý nghĩa
    - Ví dụ: sed -n 's/a/A/2pw /tmp/file' FILE

sed

- Sử dụng nhiều lệnh
  - sed -e 'lệnh 1' -e 'lệnh 2' … FILE
  - Thay vì
    - sed 's/a/A/' FILE | sed 's/b/B/'
  - Ta có thể sử dụng
    - sed -e 's/a/A/' -e 's/b/B/' FILE
- Có thể nhóm các lệnh bằng cặp dấu ngoặc {}

```
sed -n ' /begin/,/end/ {        s/#.*//
                                s/[ ^I]*$//
                                /^$/ d
                                p
} '
```

- Chú ý: ^I = ký tự TAB
- Xem thêm: http://www.grymoire.com/Unix/Sed.html

awk

- Giả sử ta có file coins.txt như sau:

```
gold     1    1986  USA                American Eagle
gold     1    1908  Austria-Hungary    Franz Josef 100 Korona
silver   10   1981  USA                ingot
gold     1    1984  Switzerland        ingot
gold     1    1979  RSA                Krugerrand
gold     0.5  1981  RSA                Krugerrand
gold     0.1  1986  PRC                Panda
silver   1    1986  USA                Liberty dollar
gold     0.25 1986  USA                Liberty 5-dollar piece
silver   0.5  1986  USA                Liberty 50-cent piece
silver   1    1987  USA                Constitution dollar
gold     0.25 1987  USA                Constitution 5-dollar piece
gold     1    1988  Canada             Maple Leaf
```

- awk '/gold/' coins.txt
  - Liệt kê các dòng có chứa từ gold

awk

- awk '/gold/ {print $5,$6,$7,$8}' coins.txt
  - Tìm những dòng chứa từ gold và in các trường 5, 6, 7, 8
  - Kết quả:
    - American Eagle
    - Franz Josef 100 Korona
    - ingot
    - Krugerrand
    - Krugerrand
    - Panda
    - Liberty 5-dollar piece
    - Constitution 5-dollar piece
    - Maple Leaf
- Cú pháp của awk là
  - awk '<mẫu> {lệnh}' FILE
  - Nếu không có <mẫu> thì phạm vi áp dụng là cả file

awk

- Lệnh print
  - In các tham số ra màn hình
  - Nếu không có tham số sẽ in toàn bộ dòng hiện hành ra màn hình, do đó
    - awk '/gold/' = awk '/gold/{print}' = awk '/gold/ {print $0}'
- $n: trường thứ n (mặc định các trường cách nhau bằng khoảng trắng, ta có thể sử dụng option -Fx để chỉ định x là ký tự phân cách)
- $0: toàn bộ dòng
- Ta có thể làm nhiều thứ phức tạp hơn nữa với awk
  - awk 'if ($3 < 1980) print $3, " ", $5, $6, $7, $8}' coins.txt
  - Kết quả:
    - 1908 Franz Josef 100 Korona
    - 1979 Krugerrand

awk

- In tổng số loại tiền đang có
  - awk 'END {print NR,"coins"}' coins.txt
  - Kết quả: **13 coins**
- Từ khóa END cho biết phần trong cặp {} chỉ thực hiện 1 lần sau khi duyệt qua hết file
- Tương tự như thế từ khóa BEGIN cho biết phần trong cặp {} chỉ thực hiện 1 lần khi bắt đầu duyệt file
- NR: tổng số dòng của file coins.txt
- NF: tổng số trường của dòng hiện hành
- Cú pháp tổng quát của awk là:

```
awk 'BEGIN {lệnh bd}
    <mẫu 1> {lệnh 1}
    <mẫu 2> {lệnh 2}
...
END {lệnh kt}'
```

awk

- Ví dụ:
  - awk '/gold/ {ounces += $2} END {print "value = $" 425*ounces}' coins.txt
- Có thể lưu các lệnh của awk trong file cmd.awk và gọi
  - awk -f cmd.awk

awk

- Ví dụ:

```
/gold/    { num_gold++; wt_gold += $2 }      # Get weight of gold.
/silver/  { num_silver++; wt_silver += $2 }  # Get weight of silver.
END { val_gold = 485 * wt_gold;              # Compute value of gold.
        val_silver = 16 * wt_silver;         # Compute value of silver.
        total = val_gold + val_silver;
        print "Summary data for coin collection:";  # Print results.
        printf ("\n");
        printf ("   Gold pieces:                %2d\n", num_gold);
        printf ("   Weight of gold pieces:      %5.2f\n", wt_gold);
        printf ("   Value of gold pieces:      %7.2f\n",val_gold);
        printf ("\n");
        printf ("   Silver pieces:              %2d\n", num_silver);
        printf ("   Weight of silver pieces:    %5.2f\n", wt_silver);
        printf ("   Value of silver pieces:    %7.2f\n",val_silver);
        printf ("\n");
        printf ("   Total number of pieces:     %2d\n", NR);
        printf ("   Value of collection:       %7.2f\n", total); }
```

awk

Summary data for coin collection:

```
Gold pieces:                9
Weight of gold pieces:      6.10
Value of gold pieces:       2958.50

Silver pieces:              4
Weight of silver pieces:    12.50
Value of silver pieces:     200.00

Total number of pieces:     13
Value of collection:        3158.50
```

awk

- Lệnh printf
  - Tương tự như printf của C
  - printf("<format_code>", <parameters>)
- Awk nâng cao
  - Cú pháp đầy đủ của awk:
    - awk [-F<ch>] {pgm} | { -f <pgm_file> } [<vars>] [- | <data_file>]
    - ch — ký tự phân cách trường (mặc định là khoảng trắng)
    - pgm — lệnh (chương trình awk)
    - pgm_file — file chứa lệnh
    - vars — danh sách các biến cần khởi tạo
    - - — sử dụng dòng nhập làm đầu vào
    - data_file — tên file đầu vào

awk

- Ví dụ trong file **sumary.awk** có đoạn

```
END { val_gold   = pg * wt_gold
  val_silver = ps * wt_silver
  ...
}
```

- Khi gọi awk ta có thể truyền 2 biến pg và ps
  - awk -f summary.awk pg=485 ps=16 coins.txt
  - Chú ý: không có khoảng trắng trước vào sau dấu =
- Mẫu:
  - Sử dụng /biểu thức chính quy/
  - Ví dụ /^The/: dòng bắt đầu bằng từ The

awk

- Mẫu (tt)
  - $1 ~ /^France$/ dòng có trường đầu tiên là France
  - $1 !~ /^Norwey$/ dòng có trường đầu tiên không phải là Norwey
  - /^Ireland/,/^Summary/ tất cả các dòng từ dòng bắt đầu bằng Ireland cho đến dòng bắt đầu bằng Summary
  - NR == 10 dòng thứ 10
  - NR == 10, NR==20 từ dòng thứ 10 đến dòng 20 (11 dòng)
  - Có thể sử dụng !=, <, <=, >, >=
    - NF == 0 dòng trống (không có trường nào)
  - Có thể sử dụng &&, || và các dấu ngoặc
    - ((NR >= 30) && ($1 == "France")) || ($1 == "Norway")

awk

- Biến
  - var = 1776 và var = "1776" là như nhau
  - var = "something" và var = something là khác nhau
  - Với var = something, (var == 0) luôn trả về true
- Biến dựng sẵn
  - NR: dòng hiện hành, khi đến cuối file NR là tổng số dòng
  - NF: số trường của dòng hiện hành
  - $1, $2, …, $NF là các trường từ 1 đến NF
  - $0: cả dòng
  - FILENAME: tên file
  - FS: ký tự ngăn cách các trường
  - RS: ký tự ngăn cách dòng (mặc định là newline)
  - OFS: ký tự ngăn cách trường của kết quả (mặc định là khoảng trắng
  - ORS: ký tự ngăn cách dòng của kết quả
- Có thể gán giá trị cho biến trường ví dụ
  - $2 = "toto"

awk

- Mảng (array)
  - Mảng chỉ số
    - ar[1], ar[2]
    - Chỉ số của phần tử đầu tiền là 1
  - Bảng băm
    - marks[“Kim”], prices[“gold”]
  - Duyệt mảng

    for (var in array)

    làm gì đó trên array[var]
  - Xóa phần tử
    - delete array[chỉ số]
  - Xóa cả mảng
    - delete array

awk

- Các phép toán
- Cấu trúc if, for, while
- Tương tự như C
  - if (<condition>) <action 1> [else <action 2>]
  - while (<condition>) <action>
- Xem thêm: http://www.vectorsite.net/tsawk.html

SHELL

- Tất cả người dùng được khai báo bằng **tài khoản + mật khẩu**
- Sau khi đăng nhập vào hệ thống, người dùng sẽ giao tiếp với hệ thống (máy tính)
- Trình thông dich cho phép người dùng giao tiếp tiếp với hệ thống LINUX gọi là SHELL
- Có nhiều trình thông dịch SHELL
  - SHELL of BOURNE (sh) của AT&T
  - Korn SHELL (ksh) trên UNIX
  - C SHELL (csh) của Berkeley
  - Tenex SHELL (tcsh)
  - **Bourne Again SHELL (bash)**

SHELL

- SHELL đóng 3 vai trò khác nhau
  - Thông dịch lệnh (giao tiếp giữa người dùng và hệ thống)
  - Tùy chọn phiên làm việc
  - Ngôn ngữ lập trình

Trình thông dịch SHELL

- Nguyên lý:
  - Vòng lặp vô tận
    - Hiển thị dấu nhắc ($) và chờ người dùng gõ lệnh
    - Sau khi người dùng ấn ENTER, SHELL sẽ đọc lệnh từ bàn phím
    - Phân tích cú pháp (kiểm tra lỗi, tách tham số, …)
    - Thay thế các ký tự đại diện/mở rộng các tham số (nếu có): SHELL Expansion
    - Thực thi lệnh
- Ví dụ:
  - SHELL hiển thị dấu nhắc $ và đọc bàn phím
  - Người dùng gõ vào **ls –l /usr**
  - SHELL tách lệnh vừa đọc thành 3 từ **ls** (tên lệnh) **-l** và **/usr** (2 tham số của lệnh ls)
  - SHELL tạo ra một tiến trình thực thi lệnh ls với 2 tham số và chờ cho đến khi tiến trình này thực hiện xong
  - Hiển thị lại dấu nhắc $ và cứ như thế, …
- Để kết thúc vòng lặp vô tận này, ta có thể gõ **exit**

Trình thông dịch SHELL

- Trích dẫn (quoting)
  - Sử dụng để loại bỏ ý nghĩa đặc biệt của 1 số từ hoặc ký tự
  - Có 3 cơ chế trích dẫn
    - Ký tự \ (escape character)
      - Bảo toàn ý nghĩa của ký tự đứng sau ****
      - Ví dụ ***** có nghĩa là ký tự * (nếu không có \, * sẽ được hiểu là ký tự mở rộng tên file)
      - Ngoại lệ: \ đứng cuối dòng có nghĩa là lệnh vẫn chưa kết thúc mà được viết tiếp ở dòng phía dưới
    - Cặp nháy đơn '…'
      - Bảo toàn ý nghĩa của từng ký tự bên trong cặp nháy đơn, dấu nháy đơn không được đặt trong cặp dấu nhay đơn
    - Cặp nháy đôi "…"
      - Bảo toàn ý nghĩa của từng ký tự bên trong cặp nháy đôi ngoại trừ $, ` và \, dấu nháy đôi có thể được đặt trong cặp dấu nháy đôi khi trước nó là \
      - Ví dụ: **echo "Holmes noi: \"Thoi ta ve\""** cho kết quả
      - **Holmes noi: "Thoi ta ve"**

Trình thông dịch SHELL

- Lệnh
  - Lệnh đơn
    - Tên lệnh và danh sách tham số cách nhau bằng khoảng trắng
    - Ví dụ: **echo Hello world**
  - Ống dẫn (pipeline) |: chuyển đầu ra của chương trình này thành đầu vào của chương trình kia
    - Ví dụ: **who | wc -l**
  - Danh sách lệnh
    - **lệnh 1; lệnh 2** (lệnh 2 thực hiện khi lệnh 1 thực hiện xong)
    - **lệnh 1 && lệnh 2** (lệnh 2 thực hiện khi lệnh 1 kết thúc trả về 0)
    - **lệnh 1 || lệnh 2** (lệnh 2 thực hiện khi lệnh 1 kết thúc trả về khác 0)
  - Lệnh phức
    - Kết hợp nhiều lệnh đơn lại tạo thành lệnh phức
    - Các cấu trúc rẽ nhánh, vòng lặp, … (xem phần sau)

Trình thông dịch SHELL

- Hàm
  - Nhóm nhiều lệnh lại với nhau
  - Cú pháp:

```
<tên hàm> () {
   Lệnh 1
   Lệnh 2
   ...
}
```

  - Ta sẽ quay lại trong phần lập trình SHELL

Trình thông dịch SHELL

- Mở rộng lệnh
  - Mở rộng với cặp dấu ngoặc {}
  - Mở rộng với dấu ~
  - Mở rộng tham số và biến
  - Thay thế lệnh
  - Mở rộng các phép toán số học
  - Mở rộng tên tập tin

Trình thông dịch SHELL

- Mở rộng với cặp dấu ngoặc {}
  - Tương tự như phép toán nhân một số với một tổng
  - Ví dụ: **echo 1{a,b,c}** cho kết quả:
  - **1a 1b 1c**
  - echo {a,b,c}{1,2,3} cho kết quả:
  - **a1 a2 a3 b1 b2 b3**
- Có thể sử dụng dấu .. khi muốn liệt kê số hoặc từng ký tự
  - **Ví du: echo {1..6}** cho kết quả:
  - **1 2 3 4 5 6**
  - **echo {1..6..2}** cho kết quả
  - **1 3 5**
  - **echo {a..d..2}** cho kết quả
  - **a d**
  - Các cặp dấu ngoặc có thể lồng nhau
  - Ví dụ: **echo {a,b{3,5}}** cho kết quả:
  - **a b3 b5**

Trình thông dịch SHELL

- Mở rộng với dấu ngã (~)
  - Tất cả các ký tự từ dấu ngã cho đến dấu / đầu tiên được xem như tên người dùng, và **~tên_người_dùng** được mở rộng thành thư mục của người dùng đó.
  - ví dụ: **~pnkhang** sẽ trở thành **/home/pnkhang**
  - Nếu giữa dấu ~ và / không có gì cả thì ~ sẽ được hiểu là $HOME
  - ~+ tương đương với $PWD
  - ~- tương đương với $OLDPWD (thư mục hiện hành trước đó)

Trình thông dịch SHELL

- Mở rộng tham số hoặc biến
  - Sử dụng dấu ${tham_số}
  - Thay thế nội dung của biến vào chỗ của ${tham_số}
  - Ví dụ
    - $1 giá trị của tham số thứ nhất
    - $2 giá trị tham số thứ 2, …

Trình thông dịch SHELL

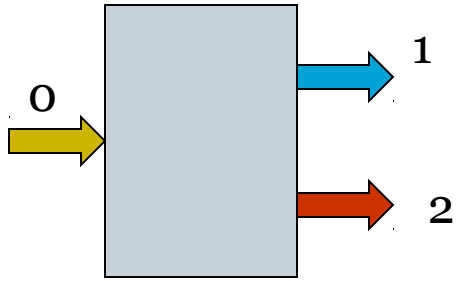
- Thay thế lệnh
  - Thay thế lệnh bằng đầu ra của nó
  - Sử dụng **`lệnh`**
  - Ví dụ **echo Tap tin /etc/passwd co `wc -l /etc/passwd`dong**
  - Ví dụ: echo Bay gio la `date`
- Tính toán biểu thức số học
  - Tính toán các phép toán trên số nguyên: +, - *, và / (chia lấy phần nguyên)
  - Sử dụng cú pháp **$((biểu thức))**
  - Ví dụ: **echo $((3 + 5))** cho kết quả **8**

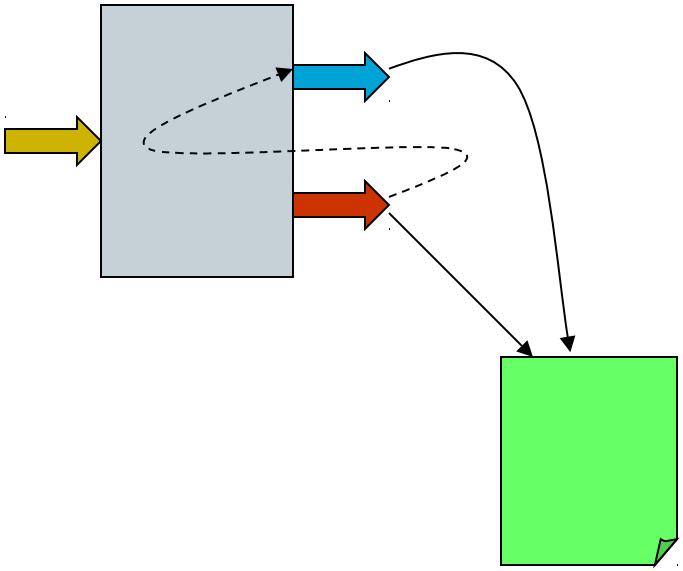
Trình thông dịch SHELL

- Mở rộng tên tập tin
  - Nếu một từ chứa "?" "*" hoặc "[" được xem như một mẫu (pattern)
- Đối sánh mẫu
  - * đại diện cho 0 hoặc nhiều ký tự bất kỳ
  - ? đại diện cho 1 ký tự bất kỳ
  - [] đại diện cho 1 trong các ký tự trọng cặp dấu ngoặc
  - [aA] a hoặc A
  - [^abc] bất cứ ký tự nào không phải là a, b hoặc c
  - [a-d] a, b, c, hoặc d
  - [a-cx-z] a, b, c, x, y, hoặc z
  - ?(danh sách mẫu) 0 hoặc 1 trong các mẫu
  - *(danh sách mẫu) 0 hoặc nhiều mẫu
  - +(danh sách mẫu) 1 hoặc nhiều mẫu
  - @(danh sách mâu) 1 trong các mẫu
  - !(danh sách mẫu) tất cả các thứ khác ngoại trừ các mẫu trong danh sách
- Chú ý: Các mẫu trong danh sách mẫu cách nhau bằng dấu |

Trình thông dịch SHELL

- Chuyển hướng các luồng nhập (0), xuất (1), lỗi (2)
  - Chuyển hướng luồng xuất: >
    - Ví dụ: **ls > list.txt**
    - Kết quả của lệnh ls (luồng xuất chuẩn) sẽ được ghi vào file list.txt
  - Chuyển hướng kép: >>
    - Ví dụ: **ls >> list.txt**
    - Kết quả của lệnh ls (luồng xuất chuẩn) sẽ được ghi **thêm** vào file list.txt
  - Chuyển hướng cả 2 luồng xuất và lỗi
    - ls > toto.txt 2>&1
    - Kết quả của lệnh ls (luồng xuất) sẽ được ghi vào file toto.txt, luồng lỗi (2>) được nối vào luồng xuất (&1) => cả 2 luồng đều được ghi vào file toto.txt
  - Chuyển hướng luồng nhập
    - **myscript < my-parameters**





Trình thông dịch SHELL

- Biến:
  - Shell sử dụng khá nhiều biến môi trường và chúng ta có thể đặt giá trị thích hợp cho các biến này để cấu hình SHELL cho phù hợp với nhu cầu của mình.
    - Ví dụ: Khi bạn gõ lệnh từ dấu nhắc của bash, SHELL sẽ tìm tập tin thực thi tương ứng trong các thư mục được đặt cho biến môi trường PATH.
  - Tên biến: bắt đầu bằng chữ cái (A-Z) hoặc gạch dưới (_), theo sau có thể là chữ cái, chữ số, hoặc gạch dưới
  - Các biến môi trường thông dụng: PATH, PWD, HOME, …

Trình thông dịch SHELL

- Gán giá trị cho biến:
  - <Biến>=<giá trị> (Không có khoảng trắng trước và sau dấu =)
  - Ví dụ: TONG=0
- Lấy giá trị của biến:
  - Đặt dấu $ trước tên biến
  - Ví dụ lệnh echo $PATH (hiển thị giá trị của biến môi trường PATH)
- Liệt kê tất cả các biến
  - set
- Xóa bỏ biến:
  - unset <Biến>

Trình thông dịch SHELL

- Một số biến định nghĩa sẵn
  - HOME thư mục người dùng
  - MAIL thư mục thư của người dùng
  - PATH danh sách các thư mục chứa lệnh (cách nhau bằng dấu hai chấm ":")
  - PS1 dấu nhắc (mặc định là "$")
  - PS1 dấu nhắc tiếp tục (khi viết lệnh trên nhiều dòng, mặc định là >)
  - USER tên người dùng
  - SHELL Shell đang sử dụng
  - PWD đường dẫn của thư mục hiện hành

Cấu hình phiên làm việc

- Tập tin cấu hình: Có 2 loại tập tin cấu hình
  - Tập tin cấu hình login:
    - **/etc/profile** và ~/.bash_profile (chạy khi login)
  - Tập tin cấu hình bash:
    - **~/.bashrc** và /etc/bashrc (nếu có) (chạy khi mở một giao dịch bash, mở một terminal chẳng hạn)
- Lệnh **export**
  - **export TÊN_BIẾN_1 TÊN_BIẾN_2**
  - Thêm các biến trên vào môi trường làm việc của các tiến trình con của SHELL hiện hành

Cấu hình phiên làm việc

- Bí danh (alias)
  - Thay thế lệnh dài bằng 1 tên khác ngắn hơn
    - Ví dụ: **alias dir='ls –l'**
    - Thực thi: **dir** (tương đượng như lệnh ls)
    - alias ls='ls –a' (định nghĩa lại)
    - dir bây giờ sẽ tương đương với **ls -l -a**
  - Xem danh sách các bí danh
    - **alias**
    - Kết quả:
      - **dir='ls -l'**
      - **ls='ls -a'**
  - Xóa bí danh
    - **unalias dir**

Cấu hình phiên làm việc

- Lịch sử lệnh (history)
  - Lệnh **history** liệt kê tất cả các lệnh được gõ
  - Có thể dùng phím mũi tên lên/xuống để quay về các lệnh trước đó

Lập trình SHELL

- Truyền tham số
  - Gọi thực thi lệnh: <lệnh> <tham số 1> <tham số 2> …
  - Để lấy giá trị tham số ta có thể sử dụng chức năng mở rộng tham số:
    - ${thứ tự của tham số}, ví dụ $1 hay ${1}, {$12}
    - Tên lệnh = $0
    - $* tất cả các tham số bắt đầu từ 1
    - $@ tương tự $*
    - $# số lượng tham số
    - $? Kết quả trả về của lệnh gần đây nhất
    - $$ pid của SHELL
  - shift [n]
    - Bỏ bớt n tham số đầu tiên kể từ 1, mặc định n = 1

Lập trình SHELL

- Gán tham số
  - Mặc định các tham số $1, $2, … $# được gán khi gọi lệnh
  - Ta có thể gán tường minh các tham số này bằng lệnh **set**
  - Ví dụ: **set a b c**
  - Ta có: **$1 = a, $2 = b, $3 = c**
  - Ví dụ: set `ls`

Lập trình SHELL

- Tập tin script
  - Nhóm các lệnh của SHELL vào tập tin => tập tin này trở thành tập tin khả thi
  - Gồm danh sách các lệnh, hàm
  - Dòng đầu tiên phải là: **#! /bin/bash**
  - Các tập tin này phải có thể đọc và thực thi (quyền phải lớn hơn 755)

Lập trình SHELL

- Truyền tham số cho script
  - Truyền tham số cho script qua dòng lệnh.
  - Lấy giá trị của tham số trong script sử dụng $<số thứ tự tham số>.
  - Ví dụ: tập tin my_cat có nội dung sau:

```
#!/bin/bash
cat $1
```

  - Để thực thi, gõ: *my_cat /etc/lilo.conf*, nội dung tập tin *lilo.conf* sẽ được hiển thị

Lập trình SHELL

- Kiểm tra biểu thức logic
  - Lệnh **test** hoặc **[**
    - Ví dụ: **test 2 = 3**
    - **echo $?** cho kết quả 1
    - **test 2 = 2**
    - **echo $?** Cho kết quả 0
  - Các phép toán kiểm tra khác
    - -a file file tồn tại
    - -b file file là 1 block file
    - -c file file là 1 character file
    - -d file file tồn tại và là 1 thư mục
    - -f file file tồn tại và là 1 tập tin
    - -h file file tồn tại và là 1 liên kết mềm
    - -r file file tồn tại và có thể đọc

Lập trình SHELL

- Các phép kiểm tra khác (tt)
  - **file1 –nt file2** file1 mới hơn file2
  - **file1 –ot file2** file1 cũ hơn

  - **-z string** true nếu chiều dài string = 0
  - **-n string** hoặc **string** true nếu chiều dài string > 0
  - string1 = string2 true nếu string1 giống string2
  - string1 != string2 true nếu string1 khác string2
  - string1 < string2
  - String1 > string2 true
  - arg1 OP arg2 so sánh hai số arg1 và arg2, OP có thể là:
    - -gt lớn hơn
    - -ge lớn hơn hoặc bằng
    - -lt nhỏ hơn
    - -le nhỏ hơn hoặc bằng
    - -eq bằng nhau
    - -ne khác

Lập trình SHELL

- Đánh giá biểu thức số học
  - Giống như C
  - let "biểu thức"
  - Ví dụ:
    - A=3
    - echo $A
    - let "A = A + 4"
    - echo $A
  - $((biểu thức))
    - Ví dụ: echo $((4 * 2))
  - expr toán_hạng OP toan_hạng
    - Ví dụ: expr 4 + 2
- Các phép toán
  - id++, id--, ++id, --id
  - +, -, *, /, %
  - **: lũy thừa
  - ~, &, |, ^: phép toán trên bit
  - <<, >>: dịch trái, phải
  - <, >, <=, >=, ==, != so sánh
  - !, &&, ||: phép toán logic
  - exp1 ? val1: val2
  - =, +=, *=, !=, …: gán
  - exp1, exp2

Lập trình SHELL

- Mảng (array)
  - Khai báo
    - <tên_biến>[chỉ số]=val1
    - Hoặc declare –a <tên_biến>
  - Gán giá trị cho biến mảng:
    - <tên_biến>=(val1 val2 … valN)
  - Truy xuất các phần tử của mảng:
    - ${ten_bien[chi_so]}
  - Số phần tử của mảng:
    - ${ten_bien[*]}
- Chú ý: Chỉ số tính từ 0

Lập trình SHELL

- Các lệnh vào, ra
  - echo "thông báo": hiển thị thông báo ra màn hình
  - echo –n "thông báo": hiển thị thông báo nhưng không xuống dòng
  - read đọc từ bàn phím và lưu giá trị vừa đọc vào biến REPLY
  - read TEN_BIEN đọc từ bàn phím và lưu giá trị vừa đọc vào biến TEN_BIEN
  - read -p "Thông báo" hiển thị thông báo và đọc từ bàn phím, lưu giá trị vào biến REPLY
  - Ví dụ:
    - read -p "Ban ten gi: " NAME
    - echo "Chao ban $NAME"

Lập trình SHELL

- Lệnh rẽ nhánh (if)

```
if lệnh kiểm tra
then
    Lệnh
fi
```

- Ví dụ:

```
if [ -f /etc/passwd ]
then
    cat /etc/passwd
fi
```

- Chú ý: điều kiện được gọi là **đúng** nếu lệnh kiểm tra trả về **0**

```
if lệnh kiểm tra
then
    Lệnh 1
else
    Lệnh 2
fi
```

```
if lệnh kiểm tra 1
then
    Lệnh 1
elif lệnh kiểm tra 2
then
    Lệnh 2
else
    Lệnh 3
fi
```

Lập trình SHELL

- Vòng lặp

```
for bien in danh_sach
do
        lệnh
done

for (( bt1; bt2; bt3 ))
do
    lệnh
done

while lệnh kiểm tra
do
    lệnh
done

until lệnh kiểm tra
do
    lệnh
done
```

- Ghi chú: vì các lệnh có thể ngăn cách vào bằng dấu ; nên ta có thể viết các lệnh gọn hơn như sau:

```
until lệnh kiểm tra ; do
    lệnh
done
```

Lập trình SHELL

- Các lệnh điều kiện dùng trong if, while, until có thể sử dụng các phép toán logic, ví dụ:
  - test -x /bin/bash && test /etc/inittab
  - [-e /bin/bash] || [-f /etc/passwd]
- Hay:
  - test -x /bin/bash -a -f /etc/inittab
  - [-e /bin/kbash -o -f /etc/passwd]
- break: thoát khỏi vòng lặp for, while, until
- continue: tiếp tục vòng lặp for, while, until

Lập trình SHELL

- Lệnh case
- Cú pháp

case biến **in**

Th1) lệnh 1;;
Th2) lệnh 2;;
...
Thn) lệnh n;;

esac

- **Ghi chú:**
  - Các trường hợp có thể là danh sách các mẫu (pattern)
  - Ví dụ:
    - *.c)
    - *.cc | *.cpp)
    - *)