================================================================

<A name=day16>

<A HREF="syllabus.html#Menu">Return to Menu</A>

Block 16

SUMMARY

READ EXCEPTIONS. EXCEPTIONS ARE AN OPTION IN C++, IN JAVA THEY ARE

MANDATORY. TAKE THE TIME TO UNDERSTAND THEM AT LEAST IN THEORY.

END SUMMARY

Exception Handling in C++

Chapter: 16.

This is really too advanced for a first semester course,

but it has some stuff in there which you should know if you

use new or delete, or if you're going on to Java. I still

prefer malloc() and free() myself, but that's the variety

of the language for you. Pretty much everything you need is

condensed into the following notefile:

Read: <A HREF="http://209.129.16.61/~hhaller/data/cisc192/modules/exceptions">exceptions</A>

Now what Exceptiona are available?

They all descend from class exception:

exception

runtime\_error

overflow\_error

underflow\_error

bad\_alloc

bad\_cast

bad\_type\_id

bad\_exception

logic\_error

invalid\_argument

length\_error

out\_of\_range

But what if you don't expect an exception, but want to be able

to handle the unexpected?

set\_unexpected

set\_terminate

Also, for pointers which automatically free what they're

pointing at when they go out of scope,

auto\_ptr< int > >ptrToInteger( new int(7 ) ) ;

// creates int ptrToInteger[7] ; which will garbage-collect

itself if it goes out of scope.

Related to assignment 7:

<A HREF="http://209.129.16.61/~hhaller/data/cisc192/modules/structio.cpp.txt">structio.cpp</A>

For Assignment 7 using C++-style I/O:

Read: <A HREF="http://209.129.16.61/~hhaller/data/cisc192/modules/structstream.cpp.txt">structstream.cpp,</A>

how to use stream methods to open, write, read, close files

Interesting:

int i ; // var not initialized

int i = int() ; // var initialized to 0

But compare int i = 0 ;

int i = int() ;

(4 more keystrokes for same result with no particular gain.)

====================================

Some comments from Richard Stallman at:

http://www.gnu.org/prep/standards/standards.html

3.1 Which Languages to Use

When you want to use a language that gets compiled and runs at high speed,

the best language to use is C. C++ is ok too, but please donâ€™t make

heavy use of templates. So is Java, if you compile it.

When highest efficiency is not required, other languages commonly used

in the free software community, such as Lisp, Scheme, Python, Ruby,

and Java, are OK too. Scheme, as implemented by GNU Guile, plays a

particular role in the GNU System: it is the preferred language to

extend programs written in C/C++, and also a fine language for a wide

range of applications. The more GNU components use Guile and Scheme,

the more users are able to extend and combine them (see The Emacs Thesis

in GNU Guile Reference Manual).

Many programs are designed to be extensible: they include an interpreter

for a language that is higher level than C. Often much of the program is

written in that language, too. The Emacs editor pioneered this technique.

The standard extensibility interpreter for GNU software is Guile

(http://www.gnu.org/software/guile/), which implements the language

Scheme (an especially clean and simple dialect of Lisp). Guile also

includes bindings for GTK+/GNOME, making it practical to write modern

GUI functionality within Guile. We donâ€™t reject programs written in

other â€œscripting languagesâ€ such as Perl and Python, but using Guile

is the path that will lead to overall consistency of the GNU system.

=========

4.2 Writing Robust Programs

Avoid arbitrary limits on the length or number of any data structure,

including file names, lines, files, and symbols, by allocating all data

structures dynamically. In most Unix utilities, â€œlong lines are silently

truncatedâ€. This is not acceptable in a GNU utility.

Utilities reading files should not drop NUL characters, or any other

nonprinting characters including those with codes above 0177. The

only sensible exceptions would be utilities specifically intended for

interface to certain types of terminals or printers that canâ€™t handle

those characters. Whenever possible, try to make programs work properly

with sequences of bytes that represent multibyte characters; UTF-8 is

the most important.

Check every system call for an error return, unless you know you wish to

ignore errors. Include the system error text (from perror, strerror, or

equivalent) in every error message resulting from a failing system call,

as well as the name of the file if any and the name of the utility. Just

â€œcannot open foo.câ€ or â€œstat failedâ€ is not sufficient.

Check every call to malloc or realloc to see if it returned zero. Check

realloc even if you are making the block smaller; in a system that rounds

block sizes to a power of 2, realloc may get a different block if you

ask for less space.

In Unix, realloc can destroy the storage block if it returns zero. GNU

realloc does not have this bug: if it fails, the original block is

unchanged. Feel free to assume the bug is fixed. If you wish to run

your program on Unix, and wish to avoid lossage in this case, you can

use the GNU malloc.

You must expect free to alter the contents of the block that was

freed. Anything you want to fetch from the block, you must fetch before

calling free.

If malloc fails in a noninteractive program, make that a fatal error. In

an interactive program (one that reads commands from the user), it is

better to abort the command and return to the command reader loop. This

allows the user to kill other processes to free up virtual memory,

and then try the command again.

Use getopt\_long to decode arguments, unless the argument syntax makes

this unreasonable.

When static storage is to be written in during program execution, use

explicit C code to initialize it. Reserve C initialized declarations

for data that will not be changed.

Try to avoid low-level interfaces to obscure Unix data structures (such

as file directories, utmp, or the layout of kernel memory), since these

are less likely to work compatibly. If you need to find all the files

in a directory, use readdir or some other high-level interface. These

are supported compatibly by GNU.

The preferred signal handling facilities are the BSD variant of signal,

and the POSIX sigaction function; the alternative USG signal interface

is an inferior design.

Nowadays, using the POSIX signal functions may be the easiest way to make

a program portable. If you use signal, then on GNU/Linux systems running

GNU libc version 1, you should include bsd/signal.h instead of signal.h,

so as to get BSD behavior. It is up to you whether to support systems

where signal has only the USG behavior, or give up on them.

In error checks that detect â€œimpossibleâ€ conditions, just abort. There

is usually no point in printing any message. These checks indicate the

existence of bugs. Whoever wants to fix the bugs will have to read the

source code and run a debugger. So explain the problem with comments in

the source. The relevant data will be in variables, which are easy to

examine with the debugger, so there is no point moving them elsewhere.

Do not use a count of errors as the exit status for a program. That does

not work, because exit status values are limited to 8 bits (0 through

255). A single run of the program might have 256 errors; if you try

to return 256 as the exit status, the parent process will see 0 as the

status, and it will appear that the program succeeded.

If you make temporary files, check the TMPDIR environment variable;

if that variable is defined, use the specified directory instead of /tmp.

In addition, be aware that there is a possible security problem when

creating temporary files in world-writable directories. In C, you can

avoid this problem by creating temporary files in this manner:

fd = open (filename, O\_WRONLY | O\_CREAT | O\_EXCL, 0600);

or by using the mkstemps function from Gnulib (see mkstemps in Gnulib).

In bash, use set -C (long name noclobber) to avoid this problem. In

addition, the mktemp utility is a more general solution for creating

temporary files from shell scripts (see mktemp invocation in GNU

Coreutils).

/\*

\* except1.cpp - demonstrate try and catch

\*/

/\*

\* except1.cpp - demo throwing and catching

\*/

#include <iostream>

using namespace std ;

class A {} ;

class B {} ;

int main(int argc, char \*argv[])

{

int x, y ;

x = 1024 ;

y = 0 ;

try {

//throw B() ;

//cerr << x << " divided by " << y << " equals: " << x/y ;

throw A() ;

}

catch(B) {

cerr << "I'm the B exception handler!\n" ;

} // catch B

catch(...) {

cerr << "I'm the general exception handler!\n" ;

return 0 ;

} // catch all others

} // main ends

#ifndef SLOPRINT\_H

#define SLOPRINT\_H

/\*-----------------------------------------------------------------

\* sloprint.h - what it sounds like

Wed Sep 5 18:45:48 PDT 2012

-----------------------------------------------------------------\*/

#include <utime.h>

void sloprint(const char \* x,int interval)

{

int i ;

for (i = 0 ; i < strlen(x) ; i++)

{

putchar(x[i]) ;

fflush(stdout) ;

usleep(interval) ;

sched\_yield() ; // "DO Events" in VB: cede rest of cpu slice

}

}

#endif

Part1 - Part2 - Part3 - Part4 - Part5 - Part6 - Part7 - MultiPage

------------------------------------------------------------------------

Unix - Frequently Asked Questions (1/7) [Frequent posting]

Not finding the answers

you need?

Submit your question for

others to see.

There are reader questions

on this topic!

Help others by sharing your

knowledge

------------------------------------------------------------------------

Message-ID:

X-Last-Updated: 1996/06/11

From: tmatimar@isgtec.com (Ted Timar)

Newsgroups: comp.unix.questions, comp.unix.shell

Subject: Unix - Frequently Asked Questions (1/7) [Frequent posting]

Date: 01 Dec 2003 09:14:51 GMT

Archive-name: unix-faq/faq/part1

Version: $Id: part1,v 2.9 1996/06/11 13:07:56 tmatimar Exp $

These seven articles contain the answers to some Frequently Asked

Questions often seen in comp.unix.questions and comp.unix.shell.

Please don't ask these questions again, they've been answered plenty

of times already - and please don't flame someone just because they may

not have read this particular posting. Thank you.

This collection of documents is Copyright (c) 1994, Ted Timar, except

Part 6, which is Copyright (c) 1994, Pierre Lewis and Ted Timar.

All rights reserved. Permission to distribute the collection is

hereby granted providing that distribution is electronic, no money

is involved, reasonable attempts are made to use the latest version

and all credits and this copyright notice are maintained.

Other requests for distribution will be considered. All reasonable

requests will be granted.

All information here has been contributed with good intentions, but

none of it is guaranteed either by the contributors or myself to be

accurate. The users of this information take all responsibility for

any damage that may occur.

Many FAQs, including this one, are available on the archive site

rtfm.mit.edu in the directory pub/usenet/news.answers.

The name under which a FAQ is archived appears in the "Archive-Name:"

line at the top of the article. This FAQ is archived as

"unix-faq/faq/part[1-7]".

These articles are divided approximately as follows:

1.\*) General questions.

2.\*) Relatively basic questions, likely to be asked by beginners.

3.\*) Intermediate questions.

4.\*) Advanced questions, likely to be asked by people who thought

they already knew all of the answers.

5.\*) Questions pertaining to the various shells, and the differences.

6.\*) An overview of Unix variants.

7.\*) An comparison of configuration management systems (RCS, SCCS).

This article includes answers to:

1.1) Who helped you put this list together?

1.2) When someone refers to 'rn(1)' or 'ctime(3)', what does

the number in parentheses mean?

1.3) What does {some strange unix command name} stand for?

1.4) How does the gateway between "comp.unix.questions" and the

"info-unix" mailing list work?

1.5) What are some useful Unix or C books?

1.6) What happened to the pronunciation list that used to be

part of this document?

If you're looking for the answer to, say, question 1.5, and want to skip

everything else, you can search ahead for the regular expression "^1.5)".

While these are all legitimate questions, they seem to crop up in

comp.unix.questions or comp.unix.shell on an annual basis, usually

followed by plenty of replies (only some of which are correct) and then

a period of griping about how the same questions keep coming up. You

may also like to read the monthly article "Answers to Frequently Asked

Questions" in the newsgroup "news.announce.newusers", which will tell

you what "UNIX" stands for.

With the variety of Unix systems in the world, it's hard to guarantee

that these answers will work everywhere. Read your local manual pages

before trying anything suggested here. If you have suggestions or

corrections for any of these answers, please send them to to

tmatimar@isgtec.com.

---------------------------------------------------------------

Subject: Who helped you put this list together?

Date: Thu Mar 18 17:16:55 EST 1993

1.1) Who helped you put this list together?

This document was one of the first collections of Frequently Asked

Questions. It was originally compiled in July 1989.

I took over the maintenance of this list. Almost all of the work

(and the credit) for generating this compilation was done by

Steve Hayman.

We also owe a great deal of thanks to dozens of Usenet readers who

submitted questions, answers, corrections and suggestions for this

list. Special thanks go to Maarten Litmaath, Guy Harris and

Jonathan Kamens, who have all made many especially valuable

contributions.

Part 5 of this document (shells) was written almost entirely by

Matthew Wicks <wicks@dcdmjw.fnal.gov>.

Part 6 of this document (Unix flavours) was written almost entirely by

Pierre (P.) Lewis <lew@bnr.ca>.

Where possible the author of each question and the date it was last

updated is given at the top. Unfortunately, I only started this

practice recently, and much of the information is lost. I was also

negligent in keeping track of who provided updates to questions.

Sorry to those who have made valuable contributions, but did not

receive the credit and recognition that they legitimately deserve.

I make this document available in \*roff format (ms and mm macro

packages). Andrew Cromarty has also converted it into Texinfo format.

Marty Leisner <leisner@sdsp.mc.xerox.com> cleaned up the Texinfo

version.

Major contributors to this document who may or may not be

recognized elsewhere are:

Steve Hayman <shayman@Objectario.com>

Pierre Lewis

Jonathan Kamens <jik@mit.edu>

Tom Christiansen <tchrist@mox.perl.com>

Maarten Litmaath <maart@nat.vu.nl>

Guy Harris <guy@auspex.com>

The formatted versions are available for anonymous ftp from

ftp.wg.omron.co.jp under pub/unix-faq/docs .

---------------------------------------------------------------

Subject: When someone refers to 'rn(1)' ... the number in parentheses mean?

Date: Tue, 13 Dec 1994 16:37:26 -0500

1.2) When someone refers to 'rn(1)' or 'ctime(3)', what does

the number in parentheses mean?

It looks like some sort of function call, but it isn't. These

numbers refer to the section of the "Unix manual" where the

appropriate documentation can be found. You could type

"man 3 ctime" to look up the manual page for "ctime" in section 3

of the manual.

The traditional manual sections are:

1 User-level commands

2 System calls

3 Library functions

4 Devices and device drivers

5 File formats

6 Games

7 Various miscellaneous stuff - macro packages etc.

8 System maintenance and operation commands

Some Unix versions use non-numeric section names. For instance,

Xenix uses "C" for commands and "S" for functions. Some newer

versions of Unix require "man -s# title" instead of "man # title".

Each section has an introduction, which you can read with "man #

intro" where # is the section number.

Sometimes the number is necessary to differentiate between a

command and a library routine or system call of the same name.

For instance, your system may have "time(1)", a manual page about

the 'time' command for timing programs, and also "time(3)", a

manual page about the 'time' subroutine for determining the

current time. You can use "man 1 time" or "man 3 time" to

specify which "time" man page you're interested in.

You'll often find other sections for local programs or even

subsections of the sections above - Ultrix has sections 3m, 3n,

3x and 3yp among others.

---------------------------------------------------------------

Subject: What does {some strange unix command name} stand for?

Date: Thu Mar 18 17:16:55 EST 1993

1.3) What does {some strange unix command name} stand for?

awk = "Aho Weinberger and Kernighan"

This language was named by its authors, Al Aho, Peter

Weinberger and Brian Kernighan.

grep = "Global Regular Expression Print"

grep comes from the ed command to print all lines matching a

certain pattern

g/re/p

where "re" is a "regular expression".

fgrep = "Fixed GREP".

fgrep searches for fixed strings only. The "f" does not stand

for "fast" - in fact, "fgrep foobar \*.c" is usually slower than

"egrep foobar \*.c" (Yes, this is kind of surprising. Try it.)

Fgrep still has its uses though, and may be useful when searching

a file for a larger number of strings than egrep can handle.

egrep = "Extended GREP"

egrep uses fancier regular expressions than grep. Many people

use egrep all the time, since it has some more sophisticated

internal algorithms than grep or fgrep, and is usually the

fastest of the three programs.

cat = "CATenate"

catenate is an obscure word meaning "to connect in a series",

which is what the "cat" command does to one or more files. Not

to be confused with C/A/T, the Computer Aided Typesetter.

gecos = "General Electric Comprehensive Operating Supervisor"

When GE's large systems division was sold to Honeywell,

Honeywell dropped the "E" from "GECOS".

Unix's password file has a "pw\_gecos" field. The name is a

real holdover from the early days. Dennis Ritchie has reported:

"Sometimes we sent printer output or batch jobs

to the GCOS machine. The gcos field in the password file

was a place to stash the information for the $IDENT card.

Not elegant."

nroff = "New ROFF"

troff = "Typesetter new ROFF"

These are descendants of "roff", which was a re-implementation

of the Multics "runoff" program (a program that you'd use to

"run off" a good copy of a document).

tee = T

From plumbing terminology for a T-shaped pipe splitter.

bss = "Block Started by Symbol"

Dennis Ritchie says:

Actually the acronym (in the sense we took it up; it may

have other credible etymologies) is "Block Started by

Symbol." It was a pseudo-op in FAP (Fortran Assembly [-er?]

Program), an assembler for the IBM 704-709-7090-7094

machines. It defined its label and set aside space for a

given number of words. There was another pseudo-op, BES,

"Block Ended by Symbol" that did the same except that the

label was defined by the last assigned word + 1. (On these

machines Fortran arrays were stored backwards in storage

and were 1-origin.)

The usage is reasonably appropriate, because just as with

standard Unix loaders, the space assigned didn't have to be

punched literally into the object deck but was represented

by a count somewhere.

biff = "BIFF"

This command, which turns on asynchronous mail notification,

was actually named after a dog at Berkeley.

I can confirm the origin of biff, if you're interested.

Biff was Heidi Stettner's dog, back when Heidi (and I, and

Bill Joy) were all grad students at U.C. Berkeley and the

early versions of BSD were being developed. Biff was

popular among the residents of Evans Hall, and was known

for barking at the mailman, hence the name of the command.

Confirmation courtesy of Eric Cooper, Carnegie Mellon University

rc (as in ".cshrc" or "/etc/rc") = "RunCom"

"rc" derives from "runcom", from the MIT CTSS system, ca. 1965.

'There was a facility that would execute a bunch of

commands stored in a file; it was called "runcom" for "run

commands", and the file began to be called "a runcom."

"rc" in Unix is a fossil from that usage.'

Brian Kernighan & Dennis Ritchie, as told to Vicki Brown

"rc" is also the name of the shell from the new Plan 9

operating system.

Perl = "Practical Extraction and Report Language"

Perl = "Pathologically Eclectic Rubbish Lister"

The Perl language is Larry Wall's highly popular

freely-available completely portable text, process, and file

manipulation tool that bridges the gap between shell and C

programming (or between doing it on the command line and

pulling your hair out). For further information, see the

Usenet newsgroup comp.lang.perl.misc.

Don Libes' book "Life with Unix" contains lots more of these

tidbits.

---------------------------------------------------------------

Subject: How does the gateway between "comp.unix.questions" ... work ?

Date: Thu Mar 18 17:16:55 EST 1993

1.4) How does the gateway between "comp.unix.questions" and the

"info-unix" mailing list work?

"info-unix" and "unix-wizards" are mailing list versions of

comp.unix.questions and comp.unix.wizards respectively.

There should be no difference in content between the

mailing list and the newsgroup.

To get on or off either of these lists, send mail to

info-unix-request@brl.mil or unix-wizards-request@brl.mil.

Be sure to use the '-Request'. Don't expect an immediate response.

Here are the gory details, courtesy of the list's maintainer,

Bob Reschly.

==== postings to info-UNIX and UNIX-wizards lists ====

Anything submitted to the list is posted; I do not moderate

incoming traffic -- BRL functions as a reflector. Postings

submitted by Internet subscribers should be addressed to the list

address (info-UNIX or UNIX- wizards); the '-request' addresses

are for correspondence with the list maintainer [me]. Postings

submitted by USENET readers should be addressed to the

appropriate news group (comp.unix.questions or

comp.unix.wizards).

For Internet subscribers, received traffic will be of two types;

individual messages, and digests. Traffic which comes to BRL

from the Internet and BITNET (via the BITNET-Internet gateway) is

immediately resent to all addressees on the mailing list.

Traffic originating on USENET is gathered up into digests which

are sent to all list members daily.

BITNET traffic is much like Internet traffic. The main

difference is that I maintain only one address for traffic

destined to all BITNET subscribers. That address points to a list

exploder which then sends copies to individual BITNET

subscribers. This way only one copy of a given message has to

cross the BITNET-Internet gateway in either direction.

USENET subscribers see only individual messages. All messages

originating on the Internet side are forwarded to our USENET

machine. They are then posted to the appropriate newsgroup.

Unfortunately, for gatewayed messages, the sender becomes

"news@brl-adm". This is currently an unavoidable side-effect of

the software which performs the gateway function.

As for readership, USENET has an extremely large readership - I

would guess several thousand hosts and tens of thousands of

readers. The master list maintained here at BRL runs about two

hundred fifty entries with roughly ten percent of those being

local redistribution lists. I don't have a good feel for the

size of the BITNET redistribution, but I would guess it is

roughly the same size and composition as the master list.

Traffic runs 150K to 400K bytes per list per week on average.

---------------------------------------------------------------

Subject: What are some useful Unix or C books?

Date: Thu Mar 18 17:16:55 EST 1993

1.5) What are some useful Unix or C books?

Mitch Wright (mitch@cirrus.com) maintains a useful list of Unix

and C books, with descriptions and some mini-reviews. There are

currently 167 titles on his list.

You can obtain a copy of this list by anonymous ftp from

ftp.rahul.net (192.160.13.1), where it's "pub/mitch/YABL/yabl".

Send additions or suggestions to mitch@cirrus.com.

Samuel Ko (kko@sfu.ca) maintains another list of Unix books.

This list contains only recommended books, and is therefore

somewhat shorter. This list is also a classified list, with

books grouped into categories, which may be better if you are

looking for a specific type of book.

You can obtain a copy of this list by anonymous ftp from

rtfm.mit.edu, where it's "pub/usenet/news.answers/books/unix".

Send additions or suggestions to kko@sfu.ca.

If you can't use anonymous ftp, email the line "help" to

"ftpmail@decwrl.dec.com" for instructions on retrieving

things via email.

---------------------------------------------------------------

Subject: What happened to the pronunciation list ... ?

Date: Thu Mar 18 17:16:55 EST 1993

1.6) What happened to the pronunciation list that used to be part of this

document?

From its inception in 1989, this FAQ document included a

comprehensive pronunciation list maintained by Maarten Litmaath

(thanks, Maarten!). It was originally created by Carl Paukstis

<carlp@frigg.isc-br.com>.

It has been retired, since it is not really relevant to the topic

of "Unix questions". You can still find it as part of the

widely-distributed "Jargon" file (maintained by Eric S. Raymond,

eric@snark.thyrsus.com) which seems like a much more appropriate

forum for the topic of "How do you pronounce /\* ?"

If you'd like a copy, you can ftp one from ftp.wg.omron.co.jp

(133.210.4.4), it's "pub/unix-faq/docs/Pronunciation-Guide".

------------------------------

End of unix/faq Digest part 1 of 7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

Ted Timar - tmatimar@isgtec.com

ISG Technologies Inc., 6509 Airport Road, Mississauga, Ontario, Canada L4V 1S7

------------------------------------------------------------------------

Part1 - Part2 - Part3 - Part4 - Part5 - Part6 - Part7 - MultiPage

Rate this FAQ

[Vote]

Current Top-Rated FAQs

Not finding the answers you need? Submit your question for others to see.

Are you an expert in this area? Share your knowledge and earn expert

points by giving answers or rating people's questions and answers! This

section of FAQS.ORG is not sanctioned in any way by FAQ authors or

maintainers.

Questions strongly related to this FAQ:

\* i\'ma newbie ,can anybody please send me an article about FreeBSD

from instalation, how do... by [-NeWbiE-] (10/16/2003)

\* what is the difference bet. kill-15, kill-9 and other kill commands

in UNIX? by hari (7/14/2003)

\* I am a beginner in UNIX, I have accidentally put a large file in my

workspace and since... by nette (8/26/2003)

\* what is the featuers of UNIX OS by frank (8/15/2003)

\* How do I find the name of the current shell that I am working

on(Whether csh or ksh or... by mulla (10/27/2003)

\* how to create an environment variable by abc (9/14/2003)

\* How to download and upload files on UNIX Server from WIN2K server???

by Arvind (10/28/2003)

\* The Linux kernel does not allow paging out of kernel memory. What

effect does this... by shida (7/20/2003)

\* Using Microport System V release 4, can the Maximum Transmission Unit

be increased to... by Donn (7/21/2003)

Questions somewhat related to this FAQ:

\* I want to know under what condition(s) the following error will be

printed? I have a... by Vidya (7/29/2003)

\* HI ! I am compiling a program where perl is embedded in C on HPUX

64 bit... by Nipun Sharma (9/26/2003)

\* I am relatively new to UNIX and C++ programming, but I want to know

can you (1).... by vhc (6/27/2003)

Other questions awaiting answers:

\* 7054 questions related to other FAQs

\* 1211 general questions

\* 2238 answered questions

---------------------------------------------------------

[ Usenet FAQs | Search | Web FAQs | Documents | RFC Index ]

---------------------------------------------------------

Send corrections/additions to the FAQ Maintainer:

tmatimar@isgtec.com (Ted Timar)

Last Update December 12 2003 @ 00:51 AM

Part1 - Part2 - Part3 - Part4 - Part5 - Part6 - Part7 - MultiPage

------------------------------------------------------------------------

Unix - Frequently Asked Questions (2/7) [Frequent posting]

Not finding the answers

you need?

Submit your question for

others to see.

There are reader questions

on this topic!

Help others by sharing your

knowledge

------------------------------------------------------------------------

Message-ID:

X-Last-Updated: 1996/06/11

From: tmatimar@isgtec.com (Ted Timar)

Newsgroups: comp.unix.questions, comp.unix.shell

Subject: Unix - Frequently Asked Questions (2/7) [Frequent posting]

Date: 01 Dec 2003 09:14:52 GMT

Archive-name: unix-faq/faq/part2

Version: $Id: part2,v 2.9 1996/06/11 13:07:56 tmatimar Exp $

These seven articles contain the answers to some Frequently Asked

Questions often seen in comp.unix.questions and comp.unix.shell.

Please don't ask these questions again, they've been answered plenty

of times already - and please don't flame someone just because they may

not have read this particular posting. Thank you.

This collection of documents is Copyright (c) 1994, Ted Timar, except

Part 6, which is Copyright (c) 1994, Pierre Lewis and Ted Timar.

All rights reserved. Permission to distribute the collection is

hereby granted providing that distribution is electronic, no money

is involved, reasonable attempts are made to use the latest version

and all credits and this copyright notice are maintained.

Other requests for distribution will be considered. All reasonable

requests will be granted.

All information here has been contributed with good intentions, but

none of it is guaranteed either by the contributors or myself to be

accurate. The users of this information take all responsibility for

any damage that may occur.

Many FAQs, including this one, are available on the archive site

rtfm.mit.edu in the directory pub/usenet/news.answers.

The name under which a FAQ is archived appears in the "Archive-Name:"

line at the top of the article. This FAQ is archived as

"unix-faq/faq/part[1-7]".

These articles are divided approximately as follows:

1.\*) General questions.

2.\*) Relatively basic questions, likely to be asked by beginners.

3.\*) Intermediate questions.

4.\*) Advanced questions, likely to be asked by people who thought

they already knew all of the answers.

5.\*) Questions pertaining to the various shells, and the differences.

6.\*) An overview of Unix variants.

7.\*) An comparison of configuration management systems (RCS, SCCS).

This article includes answers to:

2.1) How do I remove a file whose name begins with a "-" ?

2.2) How do I remove a file with funny characters in the filename ?

2.3) How do I get a recursive directory listing?

2.4) How do I get the current directory into my prompt?

2.5) How do I read characters from the terminal in a shell script?

2.6) How do I rename "\*.foo" to "\*.bar", or change file names

to lowercase?

2.7) Why do I get [some strange error message] when I

"rsh host command" ?

2.8) How do I {set an environment variable, change directory} inside a

program or shell script and have that change affect my

current shell?

2.9) How do I redirect stdout and stderr separately in csh?

2.10) How do I tell inside .cshrc if I'm a login shell?

2.11) How do I construct a shell glob-pattern that matches all files

except "." and ".." ?

2.12) How do I find the last argument in a Bourne shell script?

2.13) What's wrong with having '.' in your $PATH ?

2.14) How do I ring the terminal bell during a shell script?

2.15) Why can't I use "talk" to talk with my friend on machine X?

2.16) Why does calendar produce the wrong output?

If you're looking for the answer to, say, question 2.5, and want to skip

everything else, you can search ahead for the regular expression "^2.5)".

While these are all legitimate questions, they seem to crop up in

comp.unix.questions or comp.unix.shell on an annual basis, usually

followed by plenty of replies (only some of which are correct) and then

a period of griping about how the same questions keep coming up. You

may also like to read the monthly article "Answers to Frequently Asked

Questions" in the newsgroup "news.announce.newusers", which will tell

you what "UNIX" stands for.

With the variety of Unix systems in the world, it's hard to guarantee

that these answers will work everywhere. Read your local manual pages

before trying anything suggested here. If you have suggestions or

corrections for any of these answers, please send them to to

tmatimar@isgtec.com.

---------------------------------------------------------------

Subject: How do I remove a file whose name begins with a "-" ?

Date: Thu Mar 18 17:16:55 EST 1993

2.1) How do I remove a file whose name begins with a "-" ?

Figure out some way to name the file so that it doesn't begin

with a dash. The simplest answer is to use

rm ./-filename

(assuming "-filename" is in the current directory, of course.)

This method of avoiding the interpretation of the "-" works with

other commands too.

Many commands, particularly those that have been written to use

the "getopt(3)" argument parsing routine, accept a "--" argument

which means "this is the last option, anything after this is not

an option", so your version of rm might handle "rm -- -filename".

Some versions of rm that don't use getopt() treat a single "-"

in the same way, so you can also try "rm - -filename".

---------------------------------------------------------------

Subject: How do I remove a file with funny characters in the filename ?

Date: Thu Mar 18 17:16:55 EST 1993

2.2) How do I remove a file with funny characters in the filename ?

If the 'funny character' is a '/', skip to the last part of this

answer. If the funny character is something else, such as a ' '

or control character or character with the 8th bit set, keep reading.

The classic answers are

rm -i some\*pattern\*that\*matches\*only\*the\*file\*you\*want

which asks you whether you want to remove each file matching

the indicated pattern; depending on your shell, this may not

work if the filename has a character with the 8th bit set (the

shell may strip that off);

and

rm -ri .

which asks you whether to remove each file in the directory.

Answer "y" to the problem file and "n" to everything else.

Unfortunately this doesn't work with many versions of rm. Also

unfortunately, this will walk through every subdirectory of ".",

so you might want to "chmod a-x" those directories temporarily

to make them unsearchable.

Always take a deep breath and think about what you're doing and

double check what you typed when you use rm's "-r" flag or a

wildcard on the command line;

and

find . -type f ... -ok rm '{}' \;

where "..." is a group of predicates that uniquely identify the

file. One possibility is to figure out the inode number of the

problem file (use "ls -i .") and then use

find . -inum 12345 -ok rm '{}' \;

or

find . -inum 12345 -ok mv '{}' new-file-name \;

"-ok" is a safety check - it will prompt you for confirmation of

the command it's about to execute. You can use "-exec" instead

to avoid the prompting, if you want to live dangerously, or if

you suspect that the filename may contain a funny character

sequence that will mess up your screen when printed.

What if the filename has a '/' in it?

These files really are special cases, and can only be created by

buggy kernel code (typically by implementations of NFS that don't

filter out illegal characters in file names from remote

machines.) The first thing to do is to try to understand exactly

why this problem is so strange.

Recall that Unix directories are simply pairs of filenames and

inode numbers. A directory essentially contains information

like this:

filename inode

file1 12345

file2.c 12349

file3 12347

Theoretically, '/' and '\0' are the only two characters that

cannot appear in a filename - '/' because it's used to separate

directories and files, and '\0' because it terminates a filename.

Unfortunately some implementations of NFS will blithely create

filenames with embedded slashes in response to requests from

remote machines. For instance, this could happen when someone on

a Mac or other non-Unix machine decides to create a remote NFS

file on your Unix machine with the date in the filename. Your

Unix directory then has this in it:

filename inode

91/02/07 12357

No amount of messing around with 'find' or 'rm' as described

above will delete this file, since those utilities and all other

Unix programs, are forced to interpret the '/' in the normal way.

Any ordinary program will eventually try to do

unlink("91/02/07"), which as far as the kernel is concerned means

"unlink the file 07 in the subdirectory 02 of directory 91", but

that's not what we have - we have a \*FILE\* named "91/02/07" in

the current directory. This is a subtle but crucial distinction.

What can you do in this case? The first thing to try is to

return to the Mac that created this crummy entry, and see if you

can convince it and your local NFS daemon to rename the file to

something without slashes.

If that doesn't work or isn't possible, you'll need help from

your system manager, who will have to try the one of the

following. Use "ls -i" to find the inode number of this bogus

file, then unmount the file system and use "clri" to clear the

inode, and "fsck" the file system with your fingers crossed.

This destroys the information in the file. If you want to keep

it, you can try:

create a new directory in the same parent directory as the one

containing the bad file name;

move everything you can (i.e. everything but the file with the

bad name) from the old directory to the new one;

do "ls -id" on the directory containing the file with the bad

name to get its inumber;

umount the file system;

"clri" the directory containing the file with the bad name;

"fsck" the file system.

Then, to find the file,

remount the file system;

rename the directory you created to have the name of the old

directory (since the old directory should have been blown away

by "fsck")

move the file out of "lost+found" into the directory with a

better name.

Alternatively, you can patch the directory the hard way by

crawling around in the raw file system. Use "fsdb", if you

have it.

---------------------------------------------------------------

Subject: How do I get a recursive directory listing?

Date: Thu Mar 18 17:16:55 EST 1993

2.3) How do I get a recursive directory listing?

One of the following may do what you want:

ls -R (not all versions of "ls" have -R)

find . -print (should work everywhere)

du -a . (shows you both the name and size)

If you're looking for a wildcard pattern that will match all ".c"

files in this directory and below, you won't find one, but you

can use

% some-command `find . -name '\*.c' -print`

"find" is a powerful program. Learn about it.

---------------------------------------------------------------

Subject: How do I get the current directory into my prompt?

Date: Thu Mar 18 17:16:55 EST 1993

2.4) How do I get the current directory into my prompt?

It depends which shell you are using. It's easy with some

shells, hard or impossible with others.

C Shell (csh):

Put this in your .cshrc - customize the prompt variable the

way you want.

alias setprompt 'set prompt="${cwd}% "'

setprompt # to set the initial prompt

alias cd 'chdir \!\* && setprompt'

If you use pushd and popd, you'll also need

alias pushd 'pushd \!\* && setprompt'

alias popd 'popd \!\* && setprompt'

Some C shells don't keep a $cwd variable - you can use

`pwd` instead.

If you just want the last component of the current directory

in your prompt ("mail% " instead of "/usr/spool/mail% ")

you can use

alias setprompt 'set prompt="$cwd:t% "'

Some older csh's get the meaning of && and || reversed.

Try doing:

false && echo bug

If it prints "bug", you need to switch && and || (and get

a better version of csh.)

Bourne Shell (sh):

If you have a newer version of the Bourne Shell (SVR2 or newer)

you can use a shell function to make your own command, "xcd" say:

xcd() { cd $\* ; PS1="`pwd` $ "; }

If you have an older Bourne shell, it's complicated but not

impossible. Here's one way. Add this to your .profile file:

LOGIN\_SHELL=$$ export LOGIN\_SHELL

CMDFILE=/tmp/cd.$$ export CMDFILE

# 16 is SIGURG, pick a signal that's not likely to be used

PROMPTSIG=16 export PROMPTSIG

trap '. $CMDFILE' $PROMPTSIG

and then put this executable script (without the indentation!),

let's call it "xcd", somewhere in your PATH

: xcd directory - change directory and set prompt

: by signalling the login shell to read a command file

cat >${CMDFILE?"not set"} <

Subject: How do I read characters from the terminal in a shell script?

Date: Thu Mar 18 17:16:55 EST 1993

2.5) How do I read characters from the terminal in a shell script?

In sh, use read. It is most common to use a loop like

while read line

do

...

done

In csh, use $< like this:

while ( 1 )

set line = "$<"

if ( "$line" == "" ) break

...

end

Unfortunately csh has no way of distinguishing between a blank

line and an end-of-file.

If you're using sh and want to read a \*single\* character from the

terminal, you can try something like

echo -n "Enter a character: "

stty cbreak # or stty raw

readchar=`dd if=/dev/tty bs=1 count=1 2>/dev/null`

stty -cbreak

echo "Thank you for typing a $readchar ."

---------------------------------------------------------------

Subject: How do I rename "\*.foo" to "\*.bar", or change file names to lowercase?

Date: Thu Mar 18 17:16:55 EST 1993

2.6) How do I rename "\*.foo" to "\*.bar", or change file names to lowercase?

Why doesn't "mv \*.foo \*.bar" work? Think about how the shell

expands wildcards. "\*.foo" and "\*.bar" are expanded before the

mv command ever sees the arguments. Depending on your shell,

this can fail in a couple of ways. CSH prints "No match."

because it can't match "\*.bar". SH executes "mv a.foo b.foo

c.foo \*.bar", which will only succeed if you happen to have a

single directory named "\*.bar", which is very unlikely and almost

certainly not what you had in mind.

Depending on your shell, you can do it with a loop to "mv" each

file individually. If your system has "basename", you can use:

C Shell:

foreach f ( \*.foo )

set base=`basename $f .foo`

mv $f $base.bar

end

Bourne Shell:

for f in \*.foo; do

base=`basename $f .foo`

mv $f $base.bar

done

Some shells have their own variable substitution features, so

instead of using "basename", you can use simpler loops like:

C Shell:

foreach f ( \*.foo )

mv $f $f:r.bar

end

Korn Shell:

for f in \*.foo; do

mv $f ${f%foo}bar

done

If you don't have "basename" or want to do something like

renaming foo.\* to bar.\*, you can use something like "sed" to

strip apart the original file name in other ways, but the general

looping idea is the same. You can also convert file names into

"mv" commands with 'sed', and hand the commands off to "sh" for

execution. Try

ls -d \*.foo | sed -e 's/.\*/mv & &/' -e 's/foo$/bar/' | sh

A program by Vladimir Lanin called "mmv" that does this job

nicely was posted to comp.sources.unix (Volume 21, issues 87 and

88) in April 1990. It lets you use

mmv '\*.foo' '=1.bar'

Shell loops like the above can also be used to translate file

names from upper to lower case or vice versa. You could use

something like this to rename uppercase files to lowercase:

C Shell:

foreach f ( \* )

mv $f `echo $f | tr '[A-Z]' '[a-z]'`

end

Bourne Shell:

for f in \*; do

mv $f `echo $f | tr '[A-Z]' '[a-z]'`

done

Korn Shell:

typeset -l l

for f in \*; do

l="$f"

mv $f $l

done

If you wanted to be really thorough and handle files with `funny'

names (embedded blanks or whatever) you'd need to use

Bourne Shell:

for f in \*; do

g=`expr "xxx$f" : 'xxx\(.\*\)' | tr '[A-Z]' '[a-z]'`

mv "$f" "$g"

done

The `expr' command will always print the filename, even if it

equals `-n' or if it contains a System V escape sequence like `\c'.

Some versions of "tr" require the [ and ], some don't. It

happens to be harmless to include them in this particular

example; versions of tr that don't want the [] will conveniently

think they are supposed to translate '[' to '[' and ']' to ']'.

If you have the "perl" language installed, you may find this

rename script by Larry Wall very useful. It can be used to

accomplish a wide variety of filename changes.

#!/usr/bin/perl

#

# rename script examples from lwall:

# rename 's/\.orig$//' \*.orig

# rename 'y/A-Z/a-z/ unless /^Make/' \*

# rename '$\_ .= ".bad"' \*.f

# rename 'print "$\_: "; s/foo/bar/ if =~ /^y/i' \*

$op = shift;

for (@ARGV) {

$was = $\_;

eval $op;

die $@ if $@;

rename($was,$\_) unless $was eq $\_;

}

---------------------------------------------------------------

Subject: Why do I get [some strange error message] when I "rsh host command" ?

Date: Thu Mar 18 17:16:55 EST 1993

2.7) Why do I get [some strange error message] when I "rsh host command" ?

(We're talking about the remote shell program "rsh" or sometimes

"remsh" or "remote"; on some machines, there is a restricted shell

called "rsh", which is a different thing.)

If your remote account uses the C shell, the remote host will

fire up a C shell to execute 'command' for you, and that shell

will read your remote .cshrc file. Perhaps your .cshrc contains

a "stty", "biff" or some other command that isn't appropriate for

a non-interactive shell. The unexpected output or error message

from these commands can screw up your rsh in odd ways.

Here's an example. Suppose you have

stty erase ^H

biff y

in your .cshrc file. You'll get some odd messages like this.

% rsh some-machine date

stty: : Can't assign requested address

Where are you?

Tue Oct 1 09:24:45 EST 1991

You might also get similar errors when running certain "at" or

"cron" jobs that also read your .cshrc file.

Fortunately, the fix is simple. There are, quite possibly, a

whole \*bunch\* of operations in your ".cshrc" (e.g., "set

history=N") that are simply not worth doing except in interactive

shells. What you do is surround them in your ".cshrc" with:

if ( $?prompt ) then

operations....

endif

and, since in a non-interactive shell "prompt" won't be set, the

operations in question will only be done in interactive shells.

You may also wish to move some commands to your .login file; if

those commands only need to be done when a login session starts

up (checking for new mail, unread news and so on) it's better to

have them in the .login file.

---------------------------------------------------------------

Subject: How do I ... and have that change affect my current shell?

Date: Thu Mar 18 17:16:55 EST 1993

2.8) How do I {set an environment variable, change directory} inside

a program or shell script and have that change affect my

current shell?

In general, you can't, at least not without making special

arrangements. When a child process is created, it inherits a

copy of its parent's variables (and current directory). The

child can change these values all it wants but the changes won't

affect the parent shell, since the child is changing a copy of

the original data.

Some special arrangements are possible. Your child process could

write out the changed variables, if the parent was prepared to

read the output and interpret it as commands to set its own

variables.

Also, shells can arrange to run other shell scripts in the

context of the current shell, rather than in a child process, so

that changes will affect the original shell.

For instance, if you have a C shell script named "myscript":

cd /very/long/path

setenv PATH /something:/something-else

or the equivalent Bourne or Korn shell script

cd /very/long/path

PATH=/something:/something-else export PATH

and try to run "myscript" from your shell, your shell will fork

and run the shell script in a subprocess. The subprocess is also

running the shell; when it sees the "cd" command it changes \*its\*

current directory, and when it sees the "setenv" command it

changes \*its\* environment, but neither has any effect on the

current directory of the shell at which you're typing (your login

shell, let's say).

In order to get your login shell to execute the script (without

forking) you have to use the "." command (for the Bourne or Korn

shells) or the "source" command (for the C shell). I.e. you type

. myscript

to the Bourne or Korn shells, or

source myscript

to the C shell.

If all you are trying to do is change directory or set an

environment variable, it will probably be simpler to use a C

shell alias or Bourne/Korn shell function. See the "how do I get

the current directory into my prompt" section of this article for

some examples.

A much more detailed answer prepared by

xtm@telelogic.se (Thomas Michanek) can be found at

ftp.wg.omron.co.jp in /pub/unix-faq/docs/script-vs-env.

---------------------------------------------------------------

Subject: How do I redirect stdout and stderr separately in csh?

>From: msb@sq.com (Mark Brader)

Date: Mon, 26 Oct 1992 20:15:00 -0500

2.9) How do I redirect stdout and stderr separately in csh?

In csh, you can redirect stdout with ">", or stdout and stderr

together with ">&" but there is no direct way to redirect stderr

only. The best you can do is

( command >stdout\_file ) >&stderr\_file

which runs "command" in a subshell; stdout is redirected inside

the subshell to stdout\_file, and both stdout and stderr from the

subshell are redirected to stderr\_file, but by this point stdout

has already been redirected so only stderr actually winds up in

stderr\_file.

If what you want is to avoid redirecting stdout at all, let sh

do it for you.

sh -c 'command 2>stderr\_file'

---------------------------------------------------------------

Subject: How do I tell inside .cshrc if I'm a login shell?

Date: Thu Mar 18 17:16:55 EST 1993

2.10) How do I tell inside .cshrc if I'm a login shell?

When people ask this, they usually mean either

How can I tell if it's an interactive shell? or

How can I tell if it's a top-level shell?

You could perhaps determine if your shell truly is a login shell

(i.e. is going to source ".login" after it is done with ".cshrc")

by fooling around with "ps" and "$$". Login shells generally

have names that begin with a '-'. If you're really interested in

the other two questions, here's one way you can organize your

.cshrc to find out.

if (! $?CSHLEVEL) then

#

# This is a "top-level" shell,

# perhaps a login shell, perhaps a shell started up by

# 'rsh machine some-command'

# This is where we should set PATH and anything else we

# want to apply to every one of our shells.

#

setenv CSHLEVEL 0

set home = ~username # just to be sure

source ~/.env # environment stuff we always want

else

#

# This shell is a child of one of our other shells so

# we don't need to set all the environment variables again.

#

set tmp = $CSHLEVEL

@ tmp++

setenv CSHLEVEL $tmp

endif

# Exit from .cshrc if not interactive, e.g. under rsh

if (! $?prompt) exit

# Here we could set the prompt or aliases that would be useful

# for interactive shells only.

source ~/.aliases

---------------------------------------------------------------

Subject: How do I construct a ... matches all files except "." and ".." ?

Date: Thu Mar 18 17:16:55 EST 1993

2.11) How do I construct a shell glob-pattern that matches all files

except "." and ".." ?

You'd think this would be easy.

\* Matches all files that don't begin with a ".";

.\* Matches all files that do begin with a ".", but

this includes the special entries "." and "..",

which often you don't want;

.[!.]\* (Newer shells only; some shells use a "^" instead of

the "!"; POSIX shells must accept the "!", but may

accept a "^" as well; all portable applications shall

not use an unquoted "^" immediately following the "[")

Matches all files that begin with a "." and are

followed by a non-"."; unfortunately this will miss

"..foo";

.??\* Matches files that begin with a "." and which are

at least 3 characters long. This neatly avoids

"." and "..", but also misses ".a" .

So to match all files except "." and ".." safely you have to use

3 patterns (if you don't have filenames like ".a" you can leave

out the first):

.[!.]\* .??\* \*

Alternatively you could employ an external program or two and use

backquote substitution. This is pretty good:

`ls -a | sed -e '/^\.$/d' -e '/^\.\.$/d'`

(or `ls -A` in some Unix versions)

but even it will mess up on files with newlines, IFS characters

or wildcards in their names.

In ksh, you can use: .!(.|) \*

---------------------------------------------------------------

Subject: How do I find the last argument in a Bourne shell script?

Date: Thu Mar 18 17:16:55 EST 1993

2.12) How do I find the last argument in a Bourne shell script?

Answer by:

Martin Weitzel <@mikros.systemware.de:martin@mwtech.uucp>

Maarten Litmaath <maart@nat.vu.nl>

If you are sure the number of arguments is at most 9, you can use:

eval last=\${$#}

In POSIX-compatible shells it works for ANY number of arguments.

The following works always too:

for last

do

:

done

This can be generalized as follows:

for i

do

third\_last=$second\_last

second\_last=$last

last=$i

done

Now suppose you want to REMOVE the last argument from the list,

or REVERSE the argument list, or ACCESS the N-th argument

directly, whatever N may be. Here is a basis of how to do it,

using only built-in shell constructs, without creating subprocesses:

t0= u0= rest='1 2 3 4 5 6 7 8 9' argv=

for h in '' $rest

do

for t in "$t0" $rest

do

for u in $u0 $rest

do

case $# in

0)

break 3

esac

eval argv$h$t$u=\$1

argv="$argv \"\$argv$h$t$u\"" # (1)

shift

done

u0=0

done

t0=0

done

# now restore the arguments

eval set x "$argv" # (2)

shift

This example works for the first 999 arguments. Enough?

Take a good look at the lines marked (1) and (2) and convince

yourself that the original arguments are restored indeed, no

matter what funny characters they contain!

To find the N-th argument now you can use this:

eval argN=\$argv$N

To reverse the arguments the line marked (1) must be changed to:

argv="\"\$argv$h$t$u\" $argv"

How to remove the last argument is left as an exercise.

If you allow subprocesses as well, possibly executing nonbuilt-in

commands, the `argvN' variables can be set up more easily:

N=1

for i

do

eval argv$N=\$i

N=`expr $N + 1`

done

To reverse the arguments there is still a simpler method, that

even does not create subprocesses. This approach can also be

taken if you want to delete e.g. the last argument, but in that

case you cannot refer directly to the N-th argument any more,

because the `argvN' variables are set up in reverse order:

argv=

for i

do

eval argv$#=\$i

argv="\"\$argv$#\" $argv"

shift

done

eval set x "$argv"

shift

---------------------------------------------------------------

Subject: What's wrong with having '.' in your $PATH ?

Date: Thu Mar 18 17:16:55 EST 1993

2.13) What's wrong with having '.' in your $PATH ?

A bit of background: the PATH environment variable is a list of

directories separated by colons. When you type a command name

without giving an explicit path (e.g. you type "ls", rather than

"/bin/ls") your shell searches each directory in the PATH list in

order, looking for an executable file by that name, and the shell

will run the first matching program it finds.

One of the directories in the PATH list can be the current

directory "." . It is also permissible to use an empty directory

name in the PATH list to indicate the current directory. Both of

these are equivalent

for csh users:

setenv PATH :/usr/ucb:/bin:/usr/bin

setenv PATH .:/usr/ucb:/bin:/usr/bin

for sh or ksh users

PATH=:/usr/ucb:/bin:/usr/bin export PATH

PATH=.:/usr/ucb:/bin:/usr/bin export PATH

Having "." somewhere in the PATH is convenient - you can type

"a.out" instead of "./a.out" to run programs in the current

directory. But there's a catch.

Consider what happens in the case where "." is the first entry

in the PATH. Suppose your current directory is a publically-

writable one, such as "/tmp". If there just happens to be a

program named "/tmp/ls" left there by some other user, and you

type "ls" (intending, of course, to run the normal "/bin/ls"

program), your shell will instead run "./ls", the other user's

program. Needless to say, the results of running an unknown

program like this might surprise you.

It's slightly better to have "." at the end of the PATH:

setenv PATH /usr/ucb:/bin:/usr/bin:.

Now if you're in /tmp and you type "ls", the shell will

search /usr/ucb, /bin and /usr/bin for a program named

"ls" before it gets around to looking in ".", and there

is less risk of inadvertently running some other user's

"ls" program. This isn't 100% secure though - if you're

a clumsy typist and some day type "sl -l" instead of "ls -l",

you run the risk of running "./sl", if there is one.

Some "clever" programmer could anticipate common typing

mistakes and leave programs by those names scattered

throughout public directories. Beware.

Many seasoned Unix users get by just fine without having

"." in the PATH at all:

setenv PATH /usr/ucb:/bin:/usr/bin

If you do this, you'll need to type "./program" instead

of "program" to run programs in the current directory, but

the increase in security is probably worth it.

---------------------------------------------------------------

Subject: How do I ring the terminal bell during a shell script?

>From: uwe@mpi-sb.mpg.de (Uwe Waldmann)

Date: Fri, 30 Apr 93 16:33:00 +0200

2.14) How do I ring the terminal bell during a shell script?

The answer depends on your Unix version (or rather on the kind of

"echo" program that is available on your machine).

A BSD-like "echo" uses the "-n" option for suppressing the final

newline and does not understand the octal \nnn notation. Thus

the command is

echo -n '^G'

where ^G means a \_literal\_ BEL-character (you can produce this in

emacs using "Ctrl-Q Ctrl-G" and in vi using "Ctrl-V Ctrl-G").

A SysV-like "echo" understands the \nnn notation and uses \c to

suppress the final newline, so the answer is:

echo '\007\c'

---------------------------------------------------------------

Subject: Why can't I use "talk" to talk with my friend on machine X?

>From: tmatimar@isgtec.com (Ted Timar)

Date: Thu Mar 18 17:16:55 EST 1993

2.15) Why can't I use "talk" to talk with my friend on machine X?

Unix has three common "talk" programs, none of which can talk with

any of the others. The "old" talk accounts for the first two types.

This version (often called otalk) did not take "endian" order into

account when talking to other machines. As a consequence, the Vax

version of otalk cannot talk with the Sun version of otalk.

These versions of talk use port 517.

Around 1987, most vendors (except Sun, who took 6 years longer than

any of their competitors) standardized on a new talk (often called

ntalk) which knows about network byte order. This talk works between

all machines that have it. This version of talk uses port 518.

There are now a few talk programs that speak both ntalk and one

version of otalk. The most common of these is called "ytalk".

---------------------------------------------------------------

Subject: Why does calendar produce the wrong output?

>From: tmatimar@isgtec.com (Ted Timar)

Date: Thu Sep 8 09:45:46 EDT 1994

2.16) Why does calendar produce the wrong output?

Frequently, people find that the output for the Unix calendar

program, 'cal' produces output that they do not expect.

The calendar for September 1752 is very odd:

September 1752

S M Tu W Th F S

1 2 14 15 16

17 18 19 20 21 22 23

24 25 26 27 28 29 30

This is the month in which the US (the entire British Empire actually)

switched from the Julian to the Gregorian calendar.

The other common problem people have with the calendar program is

that they pass it arguments like 'cal 9 94'. This gives the calendar

for September of AD 94, NOT 1994.

------------------------------

End of unix/faq Digest part 2 of 7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

Ted Timar - tmatimar@isgtec.com

ISG Technologies Inc., 6509 Airport Road, Mississauga, Ontario, Canada L4V 1S7

------------------------------------------------------------------------

Part1 - Part2 - Part3 - Part4 - Part5 - Part6 - Part7 - MultiPage

Rate this FAQ

[Vote]

Current Top-Rated FAQs

Not finding the answers you need? Submit your question for others to see.

Are you an expert in this area? Share your knowledge and earn expert

points by giving answers or rating people's questions and answers! This

section of FAQS.ORG is not sanctioned in any way by FAQ authors or

maintainers.

Questions strongly related to this FAQ:

\* i\'ma newbie ,can anybody please send me an article about FreeBSD

from instalation, how do... by [-NeWbiE-] (10/16/2003)

\* what is the difference bet. kill-15, kill-9 and other kill commands

in UNIX? by hari (7/14/2003)

\* I am a beginner in UNIX, I have accidentally put a large file in my

workspace and since... by nette (8/26/2003)

\* what is the featuers of UNIX OS by frank (8/15/2003)

\* How do I find the name of the current shell that I am working

on(Whether csh or ksh or... by mulla (10/27/2003)

\* how to create an environment variable by abc (9/14/2003)

\* How to download and upload files on UNIX Server from WIN2K server???

by Arvind (10/28/2003)

\* The Linux kernel does not allow paging out of kernel memory. What

effect does this... by shida (7/20/2003)

\* Using Microport System V release 4, can the Maximum Transmission Unit

be increased to... by Donn (7/21/2003)

Questions somewhat related to this FAQ:

\* I want to know under what condition(s) the following error will be

printed? I have a... by Vidya (7/29/2003)

\* HI ! I am compiling a program where perl is embedded in C on HPUX

64 bit... by Nipun Sharma (9/26/2003)

\* I am relatively new to UNIX and C++ programming, but I want to know

can you (1).... by vhc (6/27/2003)

Other questions awaiting answers:

\* 7054 questions related to other FAQs

\* 1211 general questions

\* 2238 answered questions

---------------------------------------------------------

[ Usenet FAQs | Search | Web FAQs | Documents | RFC Index ]

---------------------------------------------------------

Send corrections/additions to the FAQ Maintainer:

tmatimar@isgtec.com (Ted Timar)

Last Update December 12 2003 @ 00:51 AM

Part1 - Part2 - Part3 - Part4 - Part5 - Part6 - Part7 - MultiPage

------------------------------------------------------------------------

Unix - Frequently Asked Questions (3/7) [Frequent posting]

Not finding the answers

you need?

Submit your question for

others to see.

There are reader questions

on this topic!

Help others by sharing your

knowledge

------------------------------------------------------------------------

Message-ID:

X-Last-Updated: 1996/06/11

From: tmatimar@isgtec.com (Ted Timar)

Newsgroups: comp.unix.questions, comp.unix.shell

Subject: Unix - Frequently Asked Questions (3/7) [Frequent posting]

Date: 01 Dec 2003 09:14:53 GMT

Archive-name: unix-faq/faq/part3

Version: $Id: part3,v 2.9 1996/06/11 13:07:56 tmatimar Exp $

These seven articles contain the answers to some Frequently Asked

Questions often seen in comp.unix.questions and comp.unix.shell.

Please don't ask these questions again, they've been answered plenty

of times already - and please don't flame someone just because they may

not have read this particular posting. Thank you.

This collection of documents is Copyright (c) 1994, Ted Timar, except

Part 6, which is Copyright (c) 1994, Pierre Lewis and Ted Timar.

All rights reserved. Permission to distribute the collection is

hereby granted providing that distribution is electronic, no money

is involved, reasonable attempts are made to use the latest version

and all credits and this copyright notice are maintained.

Other requests for distribution will be considered. All reasonable

requests will be granted.

All information here has been contributed with good intentions, but

none of it is guaranteed either by the contributors or myself to be

accurate. The users of this information take all responsibility for

any damage that may occur.

Many FAQs, including this one, are available on the archive site

rtfm.mit.edu in the directory pub/usenet/news.answers.

The name under which a FAQ is archived appears in the "Archive-Name:"

line at the top of the article. This FAQ is archived as

"unix-faq/faq/part[1-7]".

These articles are divided approximately as follows:

1.\*) General questions.

2.\*) Relatively basic questions, likely to be asked by beginners.

3.\*) Intermediate questions.

4.\*) Advanced questions, likely to be asked by people who thought

they already knew all of the answers.

5.\*) Questions pertaining to the various shells, and the differences.

6.\*) An overview of Unix variants.

7.\*) An comparison of configuration management systems (RCS, SCCS).

This article includes answers to:

3.1) How do I find the creation time of a file?

3.2) How do I use "rsh" without having the rsh hang around

until the remote command has completed?

3.3) How do I truncate a file?

3.4) Why doesn't find's "{}" symbol do what I want?

3.5) How do I set the permissions on a symbolic link?

3.6) How do I "undelete" a file?

3.7) How can a process detect if it's running in the background?

3.8) Why doesn't redirecting a loop work as intended? (Bourne shell)

3.9) How do I run 'passwd', 'ftp', 'telnet', 'tip' and other interactive

programs from a shell script or in the background?

3.10) How do I find the process ID of a program with a particular

name from inside a shell script or C program?

3.11) How do I check the exit status of a remote command

executed via "rsh" ?

3.12) Is it possible to pass shell variable settings into an awk program?

3.13) How do I get rid of zombie processes that persevere?

3.14) How do I get lines from a pipe as they are written instead of

only in larger blocks?

3.15) How do I get the date into a filename?

3.16) Why do some scripts start with #! ... ?

If you're looking for the answer to, say, question 3.5, and want to skip

everything else, you can search ahead for the regular expression "^3.5)".

While these are all legitimate questions, they seem to crop up in

comp.unix.questions or comp.unix.shell on an annual basis, usually

followed by plenty of replies (only some of which are correct) and then

a period of griping about how the same questions keep coming up. You

may also like to read the monthly article "Answers to Frequently Asked

Questions" in the newsgroup "news.announce.newusers", which will tell

you what "UNIX" stands for.

With the variety of Unix systems in the world, it's hard to guarantee

that these answers will work everywhere. Read your local manual pages

before trying anything suggested here. If you have suggestions or

corrections for any of these answers, please send them to to

tmatimar@isgtec.com.

---------------------------------------------------------------

Subject: How do I find the creation time of a file?

Date: Thu Mar 18 17:16:55 EST 1993

3.1) How do I find the creation time of a file?

You can't - it isn't stored anywhere. Files have a last-modified

time (shown by "ls -l"), a last-accessed time (shown by "ls -lu")

and an inode change time (shown by "ls -lc"). The latter is often

referred to as the "creation time" - even in some man pages -

but that's wrong; it's also set by such operations as mv, ln,

chmod, chown and chgrp.

The man page for "stat(2)" discusses this.

---------------------------------------------------------------

Subject: How do I use "rsh" without having the rsh hang around ... ?

Date: Thu Mar 18 17:16:55 EST 1993

3.2) How do I use "rsh" without having the rsh hang around until the

remote command has completed?

(See note in question 2.7 about what "rsh" we're talking about.)

The obvious answers fail:

rsh machine command &

or rsh machine 'command &'

For instance, try doing rsh machine 'sleep 60 &' and you'll see

that the 'rsh' won't exit right away. It will wait 60 seconds

until the remote 'sleep' command finishes, even though that

command was started in the background on the remote machine. So

how do you get the 'rsh' to exit immediately after the 'sleep' is

started?

The solution - if you use csh on the remote machine:

rsh machine -n 'command >&/dev/null

Subject: How do I truncate a file?

Date: Mon, 27 Mar 1995 18:09:10 -0500

3.3) How do I truncate a file?

The BSD function ftruncate() sets the length of a file.

(But not all versions behave identically.) Other Unix variants

all seem to support some version of truncation as well.

For systems which support the ftruncate function, there are

three known behaviours:

BSD 4.2 - Ultrix, SGI, LynxOS

- truncation doesn't grow file

- truncation doesn't move file pointer

BSD 4.3 - SunOS, Solaris, OSF/1, HP/UX, Amiga

- truncation can grow file

- truncation doesn't move file pointer

Cray - UniCOS 7, UniCOS 8

- truncation doesn't grow file

- truncation changes file pointer

Other systems come in four varieties:

F\_CHSIZE - Only SCO

- some systems define F\_CHSIZE but don't support it

- behaves like BSD 4.3

F\_FREESP - Only Interative Unix

- some systems (eg. Interactive Unix) define F\_FREESP but

don't support it

- behaves like BSD 4.3

chsize() - QNX and SCO

- some systems (eg. Interactive Unix) have chsize() but

don't support it

- behaves like BSD 4.3

nothing - no known systems

- there will be systems that don't support truncate at all

Moderator's Note: I grabbed the functions below a few years back.

I can no longer identify the original author.

S. Spencer Sun <spencer@ncd.com> has also

contributed a version for F\_FREESP.

functions for each non-native ftruncate follow

/\* ftruncate emulations that work on some System V's.

This file is in the public domain. \*/

#include

#include

#ifdef F\_CHSIZE

int

ftruncate (fd, length)

int fd;

off\_t length;

{

return fcntl (fd, F\_CHSIZE, length);

}

#else

#ifdef F\_FREESP

/\* The following function was written by

kucharsk@Solbourne.com (William Kucharski) \*/

#include

#include

#include

int

ftruncate (fd, length)

int fd;

off\_t length;

{

struct flock fl;

struct stat filebuf;

if (fstat (fd, &filebuf) < 0)

return -1;

if (filebuf.st\_size < length)

{

/\* Extend file length. \*/

if (lseek (fd, (length - 1), SEEK\_SET) < 0)

return -1;

/\* Write a "0" byte. \*/

if (write (fd, "", 1) != 1)

return -1;

}

else

{

/\* Truncate length. \*/

fl.l\_whence = 0;

fl.l\_len = 0;

fl.l\_start = length;

fl.l\_type = F\_WRLCK; /\* Write lock on file space. \*/

/\* This relies on the UNDOCUMENTED F\_FREESP argument to

fcntl, which truncates the file so that it ends at the

position indicated by fl.l\_start.

Will minor miracles never cease? \*/

if (fcntl (fd, F\_FREESP, &fl) < 0)

return -1;

}

return 0;

}

#else

int

ftruncate (fd, length)

int fd;

off\_t length;

{

return chsize (fd, length);

}

#endif

#endif

---------------------------------------------------------------

Subject: Why doesn't find's "{}" symbol do what I want?

Date: Thu Mar 18 17:16:55 EST 1993

3.4) Why doesn't find's "{}" symbol do what I want?

"find" has a -exec option that will execute a particular command

on all the selected files. Find will replace any "{}" it sees

with the name of the file currently under consideration.

So, some day you might try to use "find" to run a command on

every file, one directory at a time. You might try this:

find /path -type d -exec command {}/\\* \;

hoping that find will execute, in turn

command directory1/\*

command directory2/\*

...

Unfortunately, find only expands the "{}" token when it appears

by itself. Find will leave anything else like "{}/\*" alone, so

instead of doing what you want, it will do

command {}/\*

command {}/\*

...

once for each directory. This might be a bug, it might be a

feature, but we're stuck with the current behaviour.

So how do you get around this? One way would be to write a

trivial little shell script, let's say "./doit", that consists of

command "$1"/\*

You could then use

find /path -type d -exec ./doit {} \;

Or if you want to avoid the "./doit" shell script, you can use

find /path -type d -exec sh -c 'command $0/\*' {} \;

(This works because within the 'command' of "sh -c 'command' A B C ...",

$0 expands to A, $1 to B, and so on.)

or you can use the construct-a-command-with-sed trick

find /path -type d -print | sed 's:.\*:command &/\*:' | sh

If all you're trying to do is cut down on the number of times

that "command" is executed, you should see if your system has the

"xargs" command. Xargs reads arguments one line at a time from

the standard input and assembles as many of them as will fit into

one command line. You could use

find /path -print | xargs command

which would result in one or more executions of

command file1 file2 file3 file4 dir1/file1 dir1/file2

Unfortunately this is not a perfectly robust or secure solution.

Xargs expects its input lines to be terminated with newlines, so

it will be confused by files with odd characters such as newlines

in their names.

---------------------------------------------------------------

Subject: How do I set the permissions on a symbolic link?

Date: Thu Mar 18 17:16:55 EST 1993

3.5) How do I set the permissions on a symbolic link?

Permissions on a symbolic link don't really mean anything. The

only permissions that count are the permissions on the file that

the link points to.

---------------------------------------------------------------

Subject: How do I "undelete" a file?

Date: Thu Mar 18 17:16:55 EST 1993

3.6) How do I "undelete" a file?

Someday, you are going to accidentally type something like

"rm \* .foo", and find you just deleted "\*" instead of "\*.foo".

Consider it a rite of passage.

Of course, any decent systems administrator should be doing

regular backups. Check with your sysadmin to see if a recent

backup copy of your file is available. But if it isn't, read

on.

For all intents and purposes, when you delete a file with "rm" it

is gone. Once you "rm" a file, the system totally forgets which

blocks scattered around the disk were part of your file. Even

worse, the blocks from the file you just deleted are going to be

the first ones taken and scribbled upon when the system needs

more disk space. However, never say never. It is theoretically

possible \*if\* you shut down the system immediately after the "rm"

to recover portions of the data. However, you had better have a

very wizardly type person at hand with hours or days to spare to

get it all back.

Your first reaction when you "rm" a file by mistake is why not

make a shell alias or procedure which changes "rm" to move files

into a trash bin rather than delete them? That way you can

recover them if you make a mistake, and periodically clean out

your trash bin. Two points: first, this is generally accepted

as a \*bad\* idea. You will become dependent upon this behaviour

of "rm", and you will find yourself someday on a normal system

where "rm" is really "rm", and you will get yourself in trouble.

Second, you will eventually find that the hassle of dealing with

the disk space and time involved in maintaining the trash bin, it

might be easier just to be a bit more careful with "rm". For

starters, you should look up the "-i" option to "rm" in your

manual.

If you are still undaunted, then here is a possible simple

answer. You can create yourself a "can" command which moves

files into a trashcan directory. In csh(1) you can place the

following commands in the ".login" file in your home directory:

alias can 'mv \!\* ~/.trashcan' # junk file(s) to trashcan

alias mtcan 'rm -f ~/.trashcan/\*' # irretrievably empty trash

if ( ! -d ~/.trashcan ) mkdir ~/.trashcan # ensure trashcan exists

You might also want to put a:

rm -f ~/.trashcan/\*

in the ".logout" file in your home directory to automatically

empty the trash when you log out. (sh and ksh versions are left

as an exercise for the reader.)

MIT's Project Athena has produced a comprehensive

delete/undelete/expunge/purge package, which can serve as a

complete replacement for rm which allows file recovery. This

package was posted to comp.sources.misc (volume 17, issue

023-026)

---------------------------------------------------------------

Subject: How can a process detect if it's running in the background?

Date: Thu Mar 18 17:16:55 EST 1993

3.7) How can a process detect if it's running in the background?

First of all: do you want to know if you're running in the

background, or if you're running interactively? If you're

deciding whether or not you should print prompts and the like,

that's probably a better criterion. Check if standard input

is a terminal:

sh: if [ -t 0 ]; then ... fi

C: if(isatty(0)) { ... }

In general, you can't tell if you're running in the background.

The fundamental problem is that different shells and different

versions of UNIX have different notions of what "foreground" and

"background" mean - and on the most common type of system with a

better-defined notion of what they mean, programs can be moved

arbitrarily between foreground and background!

UNIX systems without job control typically put a process into the

background by ignoring SIGINT and SIGQUIT and redirecting the

standard input to "/dev/null"; this is done by the shell.

Shells that support job control, on UNIX systems that support job

control, put a process into the background by giving it a process

group ID different from the process group to which the terminal

belongs. They move it back into the foreground by setting the

terminal's process group ID to that of the process. Shells that

do \*not\* support job control, on UNIX systems that support job

control, typically do what shells do on systems that don't

support job control.

---------------------------------------------------------------

Subject: Why doesn't redirecting a loop work as intended? (Bourne shell)

Date: Thu Mar 18 17:16:55 EST 1993

3.8) Why doesn't redirecting a loop work as intended? (Bourne shell)

Take the following example:

foo=bar

while read line

do

# do something with $line

foo=bletch

done < /etc/passwd

echo "foo is now: $foo"

Despite the assignment ``foo=bletch'' this will print

``foo is now: bar'' in many implementations of the Bourne shell.

Why? Because of the following, often undocumented, feature of

historic Bourne shells: redirecting a control structure (such as

a loop, or an ``if'' statement) causes a subshell to be created,

in which the structure is executed; variables set in that

subshell (like the ``foo=bletch'' assignment) don't affect the

current shell, of course.

The POSIX 1003.2 Shell and Tools Interface standardization

committee forbids the behaviour described above, i.e. in P1003.2

conformant Bourne shells the example will print ``foo is now:

bletch''.

In historic (and P1003.2 conformant) implementations you can use

the following `trick' to get around the redirection problem:

foo=bar

# make file descriptor 9 a duplicate of file descriptor 0 (stdin);

# then connect stdin to /etc/passwd; the original stdin is now

# `remembered' in file descriptor 9; see dup(2) and sh(1)

exec 9<&0 < /etc/passwd

while read line

do

# do something with $line

foo=bletch

done

# make stdin a duplicate of file descriptor 9, i.e. reconnect

# it to the original stdin; then close file descriptor 9

exec 0<&9 9<&-

echo "foo is now: $foo"

This should always print ``foo is now: bletch''.

Right, take the next example:

foo=bar

echo bletch | read foo

echo "foo is now: $foo"

This will print ``foo is now: bar'' in many implementations,

``foo is now: bletch'' in some others. Why? Generally each part

of a pipeline is run in a different subshell; in some

implementations though, the last command in the pipeline is made

an exception: if it is a builtin command like ``read'', the

current shell will execute it, else another subshell is created.

POSIX 1003.2 allows both behaviours so portable scripts cannot

depend on any of them.

---------------------------------------------------------------

Subject: How do I run ... interactive programs from a shell script ... ?

Date: Thu Mar 18 17:16:55 EST 1993

3.9) How do I run 'passwd', 'ftp', 'telnet', 'tip' and other interactive

programs from a shell script or in the background?

These programs expect a terminal interface. Shells makes no

special provisions to provide one. Hence, such programs cannot

be automated in shell scripts.

The 'expect' program provides a programmable terminal interface

for automating interaction with such programs. The following

expect script is an example of a non-interactive version of

passwd(1).

# username is passed as 1st arg, password as 2nd

set password [index $argv 2]

spawn passwd [index $argv 1]

expect "\*password:"

send "$password\r"

expect "\*password:"

send "$password\r"

expect eof

expect can partially automate interaction which is especially

useful for telnet, rlogin, debuggers or other programs that have

no built-in command language. The distribution provides an

example script to rerun rogue until a good starting configuration

appears. Then, control is given back to the user to enjoy the game.

Fortunately some programs have been written to manage the

connection to a pseudo-tty so that you can run these sorts of

programs in a script.

To get expect, email "send pub/expect/expect.shar.Z" to

library@cme.nist.gov or anonymous ftp same from

ftp.cme.nist.gov.

Another solution is provided by the pty 4.0 program, which runs a

program under a pseudo-tty session and was posted to

comp.sources.unix, volume 25. A pty-based solution using named

pipes to do the same as the above might look like this:

#!/bin/sh

/etc/mknod out.$$ p; exec 2>&1

( exec 4/dev/null

) | ( pty passwd "$1" >out.$$ )

Here, 'waitfor' is a simple C program that searches for

its argument in the input, character by character.

A simpler pty solution (which has the drawback of not

synchronizing properly with the passwd program) is

#!/bin/sh

( sleep 5; echo "$2"; sleep 5; echo "$2") | pty passwd "$1"

---------------------------------------------------------------

Subject: How do I find the process ID of a program with a particular name ... ?

Date: Thu Mar 18 17:16:55 EST 1993

3.10) How do I find the process ID of a program with a particular name

from inside a shell script or C program?

In a shell script:

There is no utility specifically designed to map between program

names and process IDs. Furthermore, such mappings are often

unreliable, since it's possible for more than one process to have

the same name, and since it's possible for a process to change

its name once it starts running. However, a pipeline like this

can often be used to get a list of processes (owned by you) with

a particular name:

ps ux | awk '/name/ && !/awk/ {print $2}'

You replace "name" with the name of the process for which you are

searching.

The general idea is to parse the output of ps, using awk or grep

or other utilities, to search for the lines with the specified

name on them, and print the PID's for those lines. Note that the

"!/awk/" above prevents the awk process for being listed.

You may have to change the arguments to ps, depending on what

kind of Unix you are using.

In a C program:

Just as there is no utility specifically designed to map between

program names and process IDs, there are no (portable) C library

functions to do it either.

However, some vendors provide functions for reading Kernel

memory; for example, Sun provides the "kvm\_" functions, and Data

General provides the "dg\_" functions. It may be possible for any

user to use these, or they may only be useable by the super-user

(or a user in group "kmem") if read-access to kernel memory on

your system is restricted. Furthermore, these functions are

often not documented or documented badly, and might change from

release to release.

Some vendors provide a "/proc" filesystem, which appears as a

directory with a bunch of filenames in it. Each filename is a

number, corresponding to a process ID, and you can open the file

and read it to get information about the process. Once again,

access to this may be restricted, and the interface to it may

change from system to system.

If you can't use vendor-specific library functions, and you

don't have /proc, and you still want to do this completely

in C, you

are going to have to do the rummaging through kernel memory

yourself. For a good example of how to do this on many systems,

see the sources to "ofiles", available in the comp.sources.unix

archives. (A package named "kstuff" to help with kernel

rummaging was posted to alt.sources in May 1991 and is also

available via anonymous ftp as

usenet/alt.sources/articles/{329{6,7,8,9},330{0,1}}.Z from

wuarchive.wustl.edu.)

---------------------------------------------------------------

Subject: How do I check the exit status of a remote command executed via "rsh"?

Date: Thu Mar 18 17:16:55 EST 1993

3.11) How do I check the exit status of a remote command

executed via "rsh" ?

This doesn't work:

rsh some-machine some-crummy-command || echo "Command failed"

The exit status of 'rsh' is 0 (success) if the rsh program

itself completed successfully, which probably isn't what

you wanted.

If you want to check on the exit status of the remote program,

you can try using Maarten Litmaath's 'ersh' script, which was

posted to alt.sources in October 1994. ersh is a shell script

that calls rsh, arranges for the remote machine to echo the

status of the command after it completes, and exits with that

status.

---------------------------------------------------------------

Subject: Is it possible to pass shell variable settings into an awk program?

Date: Thu Mar 18 17:16:55 EST 1993

3.12) Is it possible to pass shell variable settings into an awk program?

There are two different ways to do this. The first involves

simply expanding the variable where it is needed in the program.

For example, to get a list of all ttys you're using:

who | awk '/^'"$USER"'/ { print $2 }' (1)

Single quotes are usually used to enclose awk programs because

the character '$' is often used in them, and '$' will be

interpreted by the shell if enclosed inside double quotes, but

not if enclosed inside single quotes. In this case, we \*want\*

the '$' in "$USER" to be interpreted by the shell, so we close

the single quotes and then put the "$USER" inside double quotes.

Note that there are no spaces in any of that, so the shell will

see it all as one argument. Note, further, that the double

quotes probably aren't necessary in this particular case (i.e. we

could have done

who | awk '/^'$USER'/ { print $2 }' (2)

), but they should be included nevertheless because they are

necessary when the shell variable in question contains special

characters or spaces.

The second way to pass variable settings into awk is to use an

often undocumented feature of awk which allows variable settings

to be specified as "fake file names" on the command line. For

example:

who | awk '$1 == user { print $2 }' user="$USER" - (3)

Variable settings take effect when they are encountered on the

command line, so, for example, you could instruct awk on how to

behave for different files using this technique. For example:

awk '{ program that depends on s }' s=1 file1 s=0 file2 (4)

Note that some versions of awk will cause variable settings

encountered before any real filenames to take effect before the

BEGIN block is executed, but some won't so neither way should be

relied upon.

Note, further, that when you specify a variable setting, awk

won't automatically read from stdin if no real files are

specified, so you need to add a "-" argument to the end of your

command, as I did at (3) above.

A third option is to use a newer version of awk (nawk), which allows

direct access to environment vairables. Eg.

nawk 'END { print "Your path variable is " ENVIRON["PATH"] }' /dev/null

---------------------------------------------------------------

Subject: How do I get rid of zombie processes that persevere?

>From: Jonathan I. Kamens

>From: casper@fwi.uva.nl (Casper Dik)

Date: Thu, 09 Sep 93 16:39:58 +0200

3.13) How do I get rid of zombie processes that persevere?

Unfortunately, it's impossible to generalize how the death of

child processes should behave, because the exact mechanism varies

over the various flavors of Unix.

First of all, by default, you have to do a wait() for child

processes under ALL flavors of Unix. That is, there is no flavor

of Unix that I know of that will automatically flush child

processes that exit, even if you don't do anything to tell it to

do so.

Second, under some SysV-derived systems, if you do

"signal(SIGCHLD, SIG\_IGN)" (well, actually, it may be SIGCLD

instead of SIGCHLD, but most of the newer SysV systems have

"#define SIGCHLD SIGCLD" in the header files), then child

processes will be cleaned up automatically, with no further

effort in your part. The best way to find out if it works at

your site is to try it, although if you are trying to write

portable code, it's a bad idea to rely on this in any case.

Unfortunately, POSIX doesn't allow you to do this; the behavior

of setting the SIGCHLD to SIG\_IGN under POSIX is undefined, so

you can't do it if your program is supposed to be

POSIX-compliant.

So, what's the POSIX way? As mentioned earlier, you must

install a signal handler and wait. Under POSIX signal handlers

are installed with sigaction. Since you are not interested in

``stopped'' children, only in terminated children, add SA\_NOCLDSTOP

to sa\_flags. Waiting without blocking is done with waitpid().

The first argument to waitpid should be -1 (wait for any pid),

the third should be WNOHANG. This is the most portable way

and is likely to become more portable in future.

If your systems doesn't support POSIX, there's a number of ways.

The easiest way is signal(SIGCHLD, SIG\_IGN), if it works.

If SIG\_IGN cannot be used to force automatic clean-up, then you've

got to write a signal handler to do it. It isn't easy at all to

write a signal handler that does things right on all flavors of

Unix, because of the following inconsistencies:

On some flavors of Unix, the SIGCHLD signal handler is called if

one \*or more\* children have died. This means that if your signal

handler only does one wait() call, then it won't clean up all of

the children. Fortunately, I believe that all Unix flavors for

which this is the case have available to the programmer the

wait3() or waitpid() call, which allows the WNOHANG option to

check whether or not there are any children waiting to be cleaned

up. Therefore, on any system that has wait3()/waitpid(), your

signal handler should call wait3()/waitpid() over and over again

with the WNOHANG option until there are no children left to clean

up. Waitpid() is the preferred interface, as it is in POSIX.

On SysV-derived systems, SIGCHLD signals are regenerated if there

are child processes still waiting to be cleaned up after you exit

the SIGCHLD signal handler. Therefore, it's safe on most SysV

systems to assume when the signal handler gets called that you

only have to clean up one signal, and assume that the handler

will get called again if there are more to clean up after it

exits.

On older systems, there is no way to prevent signal handlers

from being automatically reset to SIG\_DFL when the signal

handler gets called. On such systems, you have to put

"signal(SIGCHILD, catcher\_func)" (where "catcher\_func" is the

name of the handler function) as the last thing in the signal

handler, so that it gets reset.

Fortunately, newer implementations allow signal handlers to be

installed without being reset to SIG\_DFL when the handler

function is called. To get around this problem, on systems that

do not have wait3()/waitpid() but do have SIGCLD, you need to

reset the signal handler with a call to signal() after doing at

least one wait() within the handler, each time it is called. For

backward compatibility reasons, System V will keep the old

semantics (reset handler on call) of signal(). Signal handlers

that stick can be installed with sigaction() or sigset().

The summary of all this is that on systems that have waitpid()

(POSIX) or wait3(), you should use that and your signal handler

should loop, and on systems that don't, you should have one call

to wait() per invocation of the signal handler.

One more thing -- if you don't want to go through all of this

trouble, there is a portable way to avoid this problem, although

it is somewhat less efficient. Your parent process should fork,

and then wait right there and then for the child process to

terminate. The child process then forks again, giving you a

child and a grandchild. The child exits immediately (and hence

the parent waiting for it notices its death and continues to

work), and the grandchild does whatever the child was originally

supposed to. Since its parent died, it is inherited by init,

which will do whatever waiting is needed. This method is

inefficient because it requires an extra fork, but is pretty much

completely portable.

---------------------------------------------------------------

Subject: How do I get lines from a pipe ... instead of only in larger blocks?

>From: Jonathan I. Kamens

Date: Sun, 16 Feb 92 20:59:28 -0500

3.14) How do I get lines from a pipe as they are written instead of only in

larger blocks?

The stdio library does buffering differently depending on whether

it thinks it's running on a tty. If it thinks it's on a tty, it

does buffering on a per-line basis; if not, it uses a larger

buffer than one line.

If you have the source code to the client whose buffering you

want to disable, you can use setbuf() or setvbuf() to change the

buffering.

If not, the best you can do is try to convince the program that

it's running on a tty by running it under a pty, e.g. by using

the "pty" program mentioned in question 3.9.

---------------------------------------------------------------

Subject: How do I get the date into a filename?

>From: melodie neal <melodie@comtech.ct.oz.au>

Date: Fri, 7 Oct 1994 09:27:33 -0400

3.15) How do I get the date into a filename?

This isn't hard, but it is a bit cryptic at first sight. Let's

begin with the date command itself: date can take a formatting

string, to modify the way in which the date info is printed. The

formatting string has to be enclosed in quotes, to stop the shell

trying to interpret it before the date command itself gets it.

Try this:

date '+%d%m%y'

you should get back something like 130994. If you want to

punctuate this, just put the characters you would like to use in

the formatting string (NO SLASHES '/'):

date '+%d.%m.%y'

There are lots of token you can use in the formatting string:

have a look at the man page for date to find out about them.

Now, getting this into a file name. Let's say that we want to

create files called report.130994 (or whatever the date is today):

FILENAME=report.`date '+%d%m%y'`

Notice that we are using two sets of quotes here: the inner set

are to protect the formatting string from premature

interpretation; the outer set are to tell the shell to execute

the enclosed command, and substitute the result into the

expression (command substitution).

---------------------------------------------------------------

Subject: Why do some scripts start with #! ... ?

>From: chip@@chinacat.unicom.com (Chip Rosenthal)

Date: Tue, 14 Jul 1992 21:31:54 GMT

3.16) Why do some scripts start with #! ... ?

Chip Rosenthal has answered a closely related question in

comp.unix.xenix in the past.

I think what confuses people is that there exist two different

mechanisms, both spelled with the letter `#'. They both solve the

same problem over a very restricted set of cases -- but they are

none the less different.

Some background. When the UNIX kernel goes to run a program (one

of the exec() family of system calls), it takes a peek at the

first 16 bits of the file. Those 16 bits are called a `magic

number'. First, the magic number prevents the kernel from doing

something silly like trying to execute your customer database

file. If the kernel does not recognize the magic number then it

complains with an ENOEXEC error. It will execute the program only

if the magic number is recognizable.

Second, as time went on and different executable file formats were

introduced, the magic number not only told the kernel \*if\* it

could execute the file, but also \*how\* to execute the file. For

example, if you compile a program on an SCO XENIX/386 system and

carry the binary over to a SysV/386 UNIX system, the kernel will

recognize the magic number and say `Aha! This is an x.out

binary!' and configure itself to run with XENIX compatible system

calls.

Note that the kernel can only run binary executable images. So

how, you might ask, do scripts get run? After all, I can type

`my.script' at a shell prompt and I don't get an ENOEXEC error.

Script execution is done not by the kernel, but by the shell. The

code in the shell might look something like:

/\* try to run the program \*/

execl(program, basename(program), (char \*)0);

/\* the exec failed -- maybe it is a shell script? \*/

if (errno == ENOEXEC)

execl ("/bin/sh", "sh", "-c", program, (char \*)0);

/\* oh no mr bill!! \*/

perror(program);

return -1;

(This example is highly simplified. There is a lot

more involved, but this illustrates the point I'm

trying to make.)

If execl() is successful in starting the program then the code

beyond the execl() is never executed. In this example, if we can

execl() the `program' then none of the stuff beyond it is run.

Instead the system is off running the binary `program'.

If, however, the first execl() failed then this hypothetical shell

looks at why it failed. If the execl() failed because `program'

was not recognized as a binary executable, then the shell tries to

run it as a shell script.

The Berkeley folks had a neat idea to extend how the kernel starts

up programs. They hacked the kernel to recognize the magic number

`#!'. (Magic numbers are 16-bits and two 8-bit characters makes

16 bits, right?) When the `#!' magic number was recognized, the

kernel would read in the rest of the line and treat it as a

command to run upon the contents of the file. With this hack you

could now do things like:

#! /bin/sh

#! /bin/csh

#! /bin/awk -F:

This hack has existed solely in the Berkeley world, and has

migrated to USG kernels as part of System V Release 4. Prior to

V.4, unless the vendor did some special value added, the kernel

does not have the capability of doing anything other than loading

and starting a binary executable image.

Now, lets rewind a few years, to the time when more and more folks

running USG based unices were saying `/bin/sh sucks as an

interactive user interface! I want csh!'. Several vendors did

some value added magic and put csh in their distribution, even

though csh was not a part of the USG UNIX distribution.

This, however, presented a problem. Let's say you switch your

login shell to /bin/csh. Let's further suppose that you are a

cretin and insist upon programming csh scripts. You'd certainly

want to be able to type `my.script' and get it run, even though it

is a csh script. Instead of pumping it through /bin/sh, you want

the script to be started by running:

execl ("/bin/csh", "csh", "-c", "my.script", (char \*)0);

But what about all those existing scripts -- some of which are

part of the system distribution? If they started getting run by

csh then things would break. So you needed a way to run some

scripts through csh, and others through sh.

The solution introduced was to hack csh to take a look at the

first character of the script you are trying to run. If it was a

`#' then csh would try to run the script through /bin/csh,

otherwise it would run the script through /bin/sh. The example

code from the above might now look something like:

/\* try to run the program \*/

execl(program, basename(program), (char \*)0);

/\* the exec failed -- maybe it is a shell script? \*/

if (errno == ENOEXEC && (fp = fopen(program, "r")) != NULL) {

i = getc(fp);

(void) fclose(fp);

if (i == '#')

execl ("/bin/csh", "csh", "-c", program, (char \*)0);

else

execl ("/bin/sh", "sh", "-c", program, (char \*)0);

}

/\* oh no mr bill!! \*/

perror(program);

return -1;

Two important points. First, this is a `csh' hack. Nothing has

been changed in the kernel and nothing has been changed in the

other shells. If you try to execl() a script, whether or not it

begins with `#', you will still get an ENOEXEC failure. If you

try to run a script beginning with `#' from something other than

csh (e.g. /bin/sh), then it will be run by sh and not csh.

Second, the magic is that either the script begins with `#' or it

doesn't begin with `#'. What makes stuff like `:' and `: /bin/sh'

at the front of a script magic is the simple fact that they are

not `#'. Therefore, all of the following are identical at the

start of a script:

:

: /bin/sh

<--- a blank line

: /usr/games/rogue

echo "Gee...I wonder what shell I am running under???"

In all these cases, all shells will try to run the script with /bin/sh.

Similarly, all of the following are identical at the start of a script:

#

# /bin/csh

#! /bin/csh

#! /bin/sh

# Gee...I wonder what shell I am running under???

All of these start with a `#'. This means that the script will be

run by csh \*only\* if you try to start it from csh, otherwise it

will be run by /bin/sh.

(Note: if you are running ksh, substitute `ksh' for

`sh' in the above. The Korn shell is theoretically

compatible with Bourne shell, so it tries to run these

scripts itself. Your mileage may vary on some of the

other available shells such as zsh, bash, etc.)

Obviously, if you've got support for `#!' in the kernel then the

`#' hack becomes superfluous. In fact, it can be dangerous

because it creates confusion over what should happen with `#! /bin/sh'.

The `#!' handling is becoming more and more prevelant. System V

Release 4 picks up a number of the Berkeley features, including

this. Some System V Release 3.2 vendors are hacking in some of

the more visible V.4 features such as this and trying to convince

you this is sufficient and you don't need things like real,

working streams or dynamically adjustable kernel parameters.

XENIX does not support `#!'. The XENIX /bin/csh does have the `#'

hack. Support for `#!' in XENIX would be nice, but I wouldn't

hold my breath waiting for it.

------------------------------

End of unix/faq Digest part 3 of 7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

Ted Timar - tmatimar@isgtec.com

ISG Technologies Inc., 6509 Airport Road, Mississauga, Ontario, Canada L4V 1S7

------------------------------------------------------------------------

Part1 - Part2 - Part3 - Part4 - Part5 - Part6 - Part7 - MultiPage

Rate this FAQ

[Vote]

Current Top-Rated FAQs

Not finding the answers you need? Submit your question for others to see.

Are you an expert in this area? Share your knowledge and earn expert

points by giving answers or rating people's questions and answers! This

section of FAQS.ORG is not sanctioned in any way by FAQ authors or

maintainers.

Questions strongly related to this FAQ:

\* i\'ma newbie ,can anybody please send me an article about FreeBSD

from instalation, how do... by [-NeWbiE-] (10/16/2003)

\* what is the difference bet. kill-15, kill-9 and other kill commands

in UNIX? by hari (7/14/2003)

\* I am a beginner in UNIX, I have accidentally put a large file in my

workspace and since... by nette (8/26/2003)

\* what is the featuers of UNIX OS by frank (8/15/2003)

\* How do I find the name of the current shell that I am working

on(Whether csh or ksh or... by mulla (10/27/2003)

\* how to create an environment variable by abc (9/14/2003)

\* How to download and upload files on UNIX Server from WIN2K server???

by Arvind (10/28/2003)

\* The Linux kernel does not allow paging out of kernel memory. What

effect does this... by shida (7/20/2003)

\* Using Microport System V release 4, can the Maximum Transmission Unit

be increased to... by Donn (7/21/2003)

Questions somewhat related to this FAQ:

\* I want to know under what condition(s) the following error will be

printed? I have a... by Vidya (7/29/2003)

\* HI ! I am compiling a program where perl is embedded in C on HPUX

64 bit... by Nipun Sharma (9/26/2003)

\* I am relatively new to UNIX and C++ programming, but I want to know

can you (1).... by vhc (6/27/2003)

Other questions awaiting answers:

\* 7054 questions related to other FAQs

\* 1211 general questions

\* 2238 answered questions

---------------------------------------------------------

[ Usenet FAQs | Search | Web FAQs | Documents | RFC Index ]

---------------------------------------------------------

Send corrections/additions to the FAQ Maintainer:

tmatimar@isgtec.com (Ted Timar)

Last Update December 12 2003 @ 00:51 AM

Part1 - Part2 - Part3 - Part4 - Part5 - Part6 - Part7 - MultiPage

------------------------------------------------------------------------

Unix - Frequently Asked Questions (4/7) [Frequent posting]

Not finding the answers

you need?

Submit your question for

others to see.

There are reader questions

on this topic!

Help others by sharing your

knowledge

------------------------------------------------------------------------

Message-ID:

X-Last-Updated: 1996/06/11

From: tmatimar@isgtec.com (Ted Timar)

Newsgroups: comp.unix.questions, comp.unix.shell

Subject: Unix - Frequently Asked Questions (4/7) [Frequent posting]

Date: 01 Dec 2003 09:14:55 GMT

Archive-name: unix-faq/faq/part4

Version: $Id: part4,v 2.9 1996/06/11 13:07:56 tmatimar Exp $

These seven articles contain the answers to some Frequently Asked

Questions often seen in comp.unix.questions and comp.unix.shell.

Please don't ask these questions again, they've been answered plenty

of times already - and please don't flame someone just because they may

not have read this particular posting. Thank you.

This collection of documents is Copyright (c) 1994, Ted Timar, except

Part 6, which is Copyright (c) 1994, Pierre Lewis and Ted Timar.

All rights reserved. Permission to distribute the collection is

hereby granted providing that distribution is electronic, no money

is involved, reasonable attempts are made to use the latest version

and all credits and this copyright notice are maintained.

Other requests for distribution will be considered. All reasonable

requests will be granted.

All information here has been contributed with good intentions, but

none of it is guaranteed either by the contributors or myself to be

accurate. The users of this information take all responsibility for

any damage that may occur.

Many FAQs, including this one, are available on the archive site

rtfm.mit.edu in the directory pub/usenet/news.answers.

The name under which a FAQ is archived appears in the "Archive-Name:"

line at the top of the article. This FAQ is archived as

"unix-faq/faq/part[1-7]".

These articles are divided approximately as follows:

1.\*) General questions.

2.\*) Relatively basic questions, likely to be asked by beginners.

3.\*) Intermediate questions.

4.\*) Advanced questions, likely to be asked by people who thought

they already knew all of the answers.

5.\*) Questions pertaining to the various shells, and the differences.

6.\*) An overview of Unix variants.

7.\*) An comparison of configuration management systems (RCS, SCCS).

This article includes answers to:

4.1) How do I read characters from a terminal without requiring the user

to hit RETURN?

4.2) How do I check to see if there are characters to be read without

actually reading?

4.3) How do I find the name of an open file?

4.4) How can an executing program determine its own pathname?

4.5) How do I use popen() to open a process for reading AND writing?

4.6) How do I sleep() in a C program for less than one second?

4.7) How can I get setuid shell scripts to work?

4.8) How can I find out which user or process has a file open or is using

a particular file system (so that I can unmount it?)

4.9) How do I keep track of people who are fingering me?

4.10) Is it possible to reconnect a process to a terminal after it has

been disconnected, e.g. after starting a program in the background

and logging out?

4.11) Is it possible to "spy" on a terminal, displaying the output

that's appearing on it on another terminal?

If you're looking for the answer to, say, question 4.5, and want to skip

everything else, you can search ahead for the regular expression "^4.5)".

While these are all legitimate questions, they seem to crop up in

comp.unix.questions or comp.unix.shell on an annual basis, usually

followed by plenty of replies (only some of which are correct) and then

a period of griping about how the same questions keep coming up. You

may also like to read the monthly article "Answers to Frequently Asked

Questions" in the newsgroup "news.announce.newusers", which will tell

you what "UNIX" stands for.

With the variety of Unix systems in the world, it's hard to guarantee

that these answers will work everywhere. Read your local manual pages

before trying anything suggested here. If you have suggestions or

corrections for any of these answers, please send them to to

tmatimar@isgtec.com.

---------------------------------------------------------------

Subject: How do I read characters ... without requiring the user to hit RETURN?

Date: Thu Mar 18 17:16:55 EST 1993

4.1) How do I read characters from a terminal without requiring the user

to hit RETURN?

Check out cbreak mode in BSD, ~ICANON mode in SysV.

If you don't want to tackle setting the terminal parameters

yourself (using the "ioctl(2)" system call) you can let the stty

program do the work - but this is slow and inefficient, and you

should change the code to do it right some time:

#include

main()

{

int c;

printf("Hit any character to continue\n");

/\*

\* ioctl() would be better here; only lazy

\* programmers do it this way:

\*/

system("/bin/stty cbreak"); /\* or "stty raw" \*/

c = getchar();

system("/bin/stty -cbreak");

printf("Thank you for typing %c.\n", c);

exit(0);

}

Several people have sent me various more correct solutions to

this problem. I'm sorry that I'm not including any of them here,

because they really are beyond the scope of this list.

You might like to check out the documentation for the "curses"

library of portable screen functions. Often if you're interested

in single-character I/O like this, you're also interested in

doing some sort of screen display control, and the curses library

provides various portable routines for both functions.

---------------------------------------------------------------

Subject: How do I check to see if there are characters to be read ... ?

Date: Thu Mar 18 17:16:55 EST 1993

4.2) How do I check to see if there are characters to be read without

actually reading?

Certain versions of UNIX provide ways to check whether characters

are currently available to be read from a file descriptor. In

BSD, you can use select(2). You can also use the FIONREAD ioctl,

which returns the number of characters waiting to be read, but

only works on terminals, pipes and sockets. In System V Release

3, you can use poll(2), but that only works on streams. In Xenix

- and therefore Unix SysV r3.2 and later - the rdchk() system call

reports whether a read() call on a given file descriptor will block.

There is no way to check whether characters are available to be

read from a FILE pointer. (You could poke around inside stdio

data structures to see if the input buffer is nonempty, but that

wouldn't work since you'd have no way of knowing what will happen

the next time you try to fill the buffer.)

Sometimes people ask this question with the intention of writing

if (characters available from fd)

read(fd, buf, sizeof buf);

in order to get the effect of a nonblocking read. This is not

the best way to do this, because it is possible that characters

will be available when you test for availability, but will no

longer be available when you call read. Instead, set the

O\_NDELAY flag (which is also called FNDELAY under BSD) using the

F\_SETFL option of fcntl(2). Older systems (Version 7, 4.1 BSD)

don't have O\_NDELAY; on these systems the closest you can get to

a nonblocking read is to use alarm(2) to time out the read.

---------------------------------------------------------------

Subject: How do I find the name of an open file?

Date: Thu Mar 18 17:16:55 EST 1993

4.3) How do I find the name of an open file?

In general, this is too difficult. The file descriptor may

be attached to a pipe or pty, in which case it has no name.

It may be attached to a file that has been removed. It may

have multiple names, due to either hard or symbolic links.

If you really need to do this, and be sure you think long

and hard about it and have decided that you have no choice,

you can use find with the -inum and possibly -xdev option,

or you can use ncheck, or you can recreate the functionality

of one of these within your program. Just realize that

searching a 600 megabyte filesystem for a file that may not

even exist is going to take some time.

---------------------------------------------------------------

Subject: How can an executing program determine its own pathname?

Date: Thu Mar 18 17:16:55 EST 1993

4.4) How can an executing program determine its own pathname?

Your program can look at argv[0]; if it begins with a "/", it is

probably the absolute pathname to your program, otherwise your

program can look at every directory named in the environment

variable PATH and try to find the first one that contains an

executable file whose name matches your program's argv[0] (which

by convention is the name of the file being executed). By

concatenating that directory and the value of argv[0] you'd

probably have the right name.

You can't really be sure though, since it is quite legal for one

program to exec() another with any value of argv[0] it desires.

It is merely a convention that new programs are exec'd with the

executable file name in argv[0].

For instance, purely a hypothetical example:

#include

main()

{

execl("/usr/games/rogue", "vi Thesis", (char \*)NULL);

}

The executed program thinks its name (its argv[0] value) is

"vi Thesis". (Certain other programs might also think that

the name of the program you're currently running is "vi Thesis",

but of course this is just a hypothetical example, don't

try it yourself :-)

---------------------------------------------------------------

Subject: How do I use popen() to open a process for reading AND writing?

Date: Thu Mar 18 17:16:55 EST 1993

4.5) How do I use popen() to open a process for reading AND writing?

The problem with trying to pipe both input and output to an

arbitrary slave process is that deadlock can occur, if both

processes are waiting for not-yet-generated input at the same

time. Deadlock can be avoided only by having BOTH sides follow a

strict deadlock-free protocol, but since that requires

cooperation from the processes it is inappropriate for a

popen()-like library function.

The 'expect' distribution includes a library of functions that a

C programmer can call directly. One of the functions does the

equivalent of a popen for both reading and writing. It uses ptys

rather than pipes, and has no deadlock problem. It's portable to

both BSD and SV. See question 3.9 for more about 'expect'.

---------------------------------------------------------------

Subject: How do I sleep() in a C program for less than one second?

Date: Thu Mar 18 17:16:55 EST 1993

4.6) How do I sleep() in a C program for less than one second?

The first thing you need to be aware of is that all you can

specify is a MINIMUM amount of delay; the actual delay will

depend on scheduling issues such as system load, and could be

arbitrarily large if you're unlucky.

There is no standard library function that you can count on in

all environments for "napping" (the usual name for short

sleeps). Some environments supply a "usleep(n)" function which

suspends execution for n microseconds. If your environment

doesn't support usleep(), here are a couple of implementations

for BSD and System V environments.

The following code is adapted from Doug Gwyn's System V emulation

support for 4BSD and exploits the 4BSD select() system call.

Doug originally called it 'nap()'; you probably want to call it

"usleep()";

/\*

usleep -- support routine for 4.2BSD system call emulations

last edit: 29-Oct-1984 D A Gwyn

\*/

extern int select();

int

usleep( usec ) /\* returns 0 if ok, else -1 \*/

long usec; /\* delay in microseconds \*/

{

static struct /\* `timeval' \*/

{

long tv\_sec; /\* seconds \*/

long tv\_usec; /\* microsecs \*/

} delay; /\* \_select() timeout \*/

delay.tv\_sec = usec / 1000000L;

delay.tv\_usec = usec % 1000000L;

return select( 0, (long \*)0, (long \*)0, (long \*)0, &delay );

}

On System V you might do it this way:

/\*

subseconds sleeps for System V - or anything that has poll()

Don Libes, 4/1/1991

The BSD analog to this function is defined in terms of

microseconds while poll() is defined in terms of milliseconds.

For compatibility, this function provides accuracy "over the long

run" by truncating actual requests to milliseconds and

accumulating microseconds across calls with the idea that you are

probably calling it in a tight loop, and that over the long run,

the error will even out.

If you aren't calling it in a tight loop, then you almost

certainly aren't making microsecond-resolution requests anyway,

in which case you don't care about microseconds. And if you did,

you wouldn't be using UNIX anyway because random system

indigestion (i.e., scheduling) can make mincemeat out of any

timing code.

Returns 0 if successful timeout, -1 if unsuccessful.

\*/

#include

int

usleep(usec)

unsigned int usec; /\* microseconds \*/

{

static subtotal = 0; /\* microseconds \*/

int msec; /\* milliseconds \*/

/\* 'foo' is only here because some versions of 5.3 have

\* a bug where the first argument to poll() is checked

\* for a valid memory address even if the second argument is 0.

\*/

struct pollfd foo;

subtotal += usec;

/\* if less then 1 msec request, do nothing but remember it \*/

if (subtotal < 1000) return(0);

msec = subtotal/1000;

subtotal = subtotal%1000;

return poll(&foo,(unsigned long)0,msec);

}

Another possibility for nap()ing on System V, and probably other

non-BSD Unices is Jon Zeeff's s5nap package, posted to

comp.sources.misc, volume 4. It does require a installing a

device driver, but works flawlessly once installed. (Its

resolution is limited to the kernel HZ value, since it uses the

kernel delay() routine.)

Many newer versions of Unix have a nanosleep function.

---------------------------------------------------------------

Subject: How can I get setuid shell scripts to work?

Date: Thu Mar 18 17:16:55 EST 1993

4.7) How can I get setuid shell scripts to work?

[ This is a long answer, but it's a complicated and frequently-asked

question. Thanks to Maarten Litmaath for this answer, and

for the "indir" program mentioned below. ]

Let us first assume you are on a UNIX variant (e.g. 4.3BSD or

SunOS) that knows about so-called `executable shell scripts'.

Such a script must start with a line like:

#!/bin/sh

The script is called `executable' because just like a real (binary)

executable it starts with a so-called `magic number' indicating

the type of the executable. In our case this number is `#!' and

the OS takes the rest of the first line as the interpreter for

the script, possibly followed by 1 initial option like:

#!/bin/sed -f

Suppose this script is called `foo' and is found in /bin,

then if you type:

foo arg1 arg2 arg3

the OS will rearrange things as though you had typed:

/bin/sed -f /bin/foo arg1 arg2 arg3

There is one difference though: if the setuid permission bit for

`foo' is set, it will be honored in the first form of the

command; if you really type the second form, the OS will honor

the permission bits of /bin/sed, which is not setuid, of course.

----------

OK, but what if my shell script does NOT start with such a `#!'

line or my OS does not know about it?

Well, if the shell (or anybody else) tries to execute it, the OS

will return an error indication, as the file does not start with

a valid magic number. Upon receiving this indication the shell

ASSUMES the file to be a shell script and gives it another try:

/bin/sh shell\_script arguments

But we have already seen that a setuid bit on `shell\_script' will

NOT be honored in this case!

----------

Right, but what about the security risks of setuid shell scripts?

Well, suppose the script is called `/etc/setuid\_script', starting

with:

#!/bin/sh

Now let us see what happens if we issue the following commands:

$ cd /tmp

$ ln /etc/setuid\_script -i

$ PATH=.

$ -i

We know the last command will be rearranged to:

/bin/sh -i

But this command will give us an interactive shell, setuid to the

owner of the script!

Fortunately this security hole can easily be closed by making the

first line:

#!/bin/sh -

The `-' signals the end of the option list: the next argument `-i'

will be taken as the name of the file to read commands from, just

like it should!

---------

There are more serious problems though:

$ cd /tmp

$ ln /etc/setuid\_script temp

$ nice -20 temp &

$ mv my\_script temp

The third command will be rearranged to:

nice -20 /bin/sh - temp

As this command runs so slowly, the fourth command might be able

to replace the original `temp' with `my\_script' BEFORE `temp' is

opened by the shell! There are 4 ways to fix this security hole:

1) let the OS start setuid scripts in a different, secure way

- System V R4 and 4.4BSD use the /dev/fd driver to pass the

interpreter a file descriptor for the script

2) let the script be interpreted indirectly, through a frontend

that makes sure everything is all right before starting the

real interpreter - if you use the `indir' program from

comp.sources.unix the setuid script will look like this:

#!/bin/indir -u

#?/bin/sh /etc/setuid\_script

3) make a `binary wrapper': a real executable that is setuid and

whose only task is to execute the interpreter with the name of

the script as an argument

4) make a general `setuid script server' that tries to locate the

requested `service' in a database of valid scripts and upon

success will start the right interpreter with the right

arguments.

---------

Now that we have made sure the right file gets interpreted, are

there any risks left?

Certainly! For shell scripts you must not forget to set the PATH

variable to a safe path explicitly. Can you figure out why?

Also there is the IFS variable that might cause trouble if not

set properly. Other environment variables might turn out to

compromise security as well, e.g. SHELL... Furthermore you must

make sure the commands in the script do not allow interactive

shell escapes! Then there is the umask which may have been set

to something strange...

Etcetera. You should realise that a setuid script `inherits' all

the bugs and security risks of the commands that it calls!

All in all we get the impression setuid shell scripts are quite a

risky business! You may be better off writing a C program instead!

---------------------------------------------------------------

Subject: How can I find out which user or process has a file open ... ?

Date: Thu Mar 18 17:16:55 EST 1993

4.8) How can I find out which user or process has a file open or is using

a particular file system (so that I can unmount it?)

Use fuser (system V), fstat (BSD), ofiles (public domain) or

pff (public domain). These programs will tell you various things

about processes using particular files.

A port of the 4.3 BSD fstat to Dynix, SunOS and Ultrix

can be found in archives of comp.sources.unix, volume 18.

pff is part of the kstuff package, and works on quite a few systems.

Instructions for obtaining kstuff are provided in question 3.10.

I've been informed that there is also a program called lsof. I

don't know where it can be obtained.

Michael Fink <Michael.Fink@uibk.ac.at> adds:

If you are unable to unmount a file system for which above tools

do not report any open files make sure that the file system that

you are trying to unmount does not contain any active mount

points (df(1)).

---------------------------------------------------------------

Subject: How do I keep track of people who are fingering me?

>From: Jonathan I. Kamens

>From: malenovi@plains.NoDak.edu (Nikola Malenovic)

Date: Thu, 29 Sep 1994 07:28:37 -0400

4.9) How do I keep track of people who are fingering me?

Generally, you can't find out the userid of someone who is

fingering you from a remote machine. You may be able to

find out which machine the remote request is coming from.

One possibility, if your system supports it and assuming

the finger daemon doesn't object, is to make your .plan file a

"named pipe" instead of a plain file. (Use 'mknod' to do this.)

You can then start up a program that will open your .plan file

for writing; the open will block until some other process (namely

fingerd) opens the .plan for reading. Now you can feed whatever you

want through this pipe, which lets you show different .plan

information every time someone fingers you. One program for

doing this is the "planner" package in volume 41 of the

comp.sources.misc archives.

Of course, this may not work at all if your system doesn't

support named pipes or if your local fingerd insists

on having plain .plan files.

Your program can also take the opportunity to look at the output

of "netstat" and spot where an incoming finger connection is

coming from, but this won't get you the remote user.

Getting the remote userid would require that the remote site be

running an identity service such as RFC 931. There are now three

RFC 931 implementations for popular BSD machines, and several

applications (such as the wuarchive ftpd) supporting the server.

For more information join the rfc931-users mailing list,

>rfc931-users-request@kramden.acf.nyu.edu.

There are three caveats relating to this answer. The first is

that many NFS systems won't recognize the named pipe correctly.

This means that trying to read the pipe on another machine will

either block until it times out, or see it as a zero-length file,

and never print it.

The second problem is that on many systems, fingerd checks that

the .plan file contains data (and is readable) before trying to

read it. This will cause remote fingers to miss your .plan file

entirely.

The third problem is that a system that supports named pipes

usually has a fixed number of named pipes available on the

system at any given time - check the kernel config file and

FIFOCNT option. If the number of pipes on the system exceeds the

FIFOCNT value, the system blocks new pipes until somebody frees

the resources. The reason for this is that buffers are allocated

in a non-paged memory.

---------------------------------------------------------------

Subject: Is it possible to reconnect a process to a terminal ... ?

Date: Thu Mar 18 17:16:55 EST 1993

4.10) Is it possible to reconnect a process to a terminal after it has

been disconnected, e.g. after starting a program in the background

and logging out?

Most variants of Unix do not support "detaching" and "attaching"

processes, as operating systems such as VMS and Multics support.

However, there are three freely redistributable packages which can

be used to start processes in such a way that they can be later

reattached to a terminal.

The first is "screen," which is described in the

comp.sources.unix archives as "Screen, multiple windows on a CRT"

(see the "screen-3.2" package in comp.sources.misc, volume 28.)

This package will run on at least BSD, System V r3.2 and SCO UNIX.

The second is "pty," which is described in the comp.sources.unix

archives as a package to "Run a program under a pty session" (see

"pty" in volume 23). pty is designed for use under BSD-like

system only.

The third is "dislocate," which is a script that comes with the

expect distribution. Unlike the previous two, this should run on

all UNIX versions. Details on getting expect can be found in

question 3.9 .

None of these packages is retroactive, i.e. you must have

started a process under screen or pty in order to be able to

detach and reattach it.

---------------------------------------------------------------

Subject: Is it possible to "spy" on a terminal ... ?

Date: Wed, 28 Dec 1994 18:35:00 -0500

4.11) Is it possible to "spy" on a terminal, displaying the output

that's appearing on it on another terminal?

There are a few different ways you can do this, although none

of them is perfect:

\* kibitz allows two (or more) people to interact with a shell

(or any arbitary program). Uses include:

- watching or aiding another person's terminal session;

- recording a conversation while retaining the ability to

scroll backwards, save the conversation, or even edit it

while in progress;

- teaming up on games, document editing, or other cooperative

tasks where each person has strengths and weakness that

complement one another.

kibitz comes as part of the expect distribution. See question 3.9.

kibitz requires permission from the person to be spyed upon. To

spy without permission requires less pleasant approaches:

\* You can write a program that rummages through Kernel structures

and watches the output buffer for the terminal in question,

displaying characters as they are output. This, obviously, is

not something that should be attempted by anyone who does not

have experience working with the Unix kernel. Furthermore,

whatever method you come up with will probably be quite

non-portable.

\* If you want to do this to a particular hard-wired terminal all

the time (e.g. if you want operators to be able to check the

console terminal of a machine from other machines), you can

actually splice a monitor into the cable for the terminal. For

example, plug the monitor output into another machine's serial

port, and run a program on that port that stores its input

somewhere and then transmits it out \*another\* port, this one

really going to the physical terminal. If you do this, you have

to make sure that any output from the terminal is transmitted

back over the wire, although if you splice only into the

computer->terminal wires, this isn't much of a problem. This is

not something that should be attempted by anyone who is not very

familiar with terminal wiring and such.

\* The latest version of screen includes a multi-user mode.

Some details about screen can be found in question 4.10.

\* If the system being used has streams (SunOS, SVR4), the advise

program that was posted in volume 28 of comp.sources.misc can

be used. AND it doesn't requirethat it be run first (you do

have to configure your system in advance to automatically push

the advise module on the stream whenever a tty or pty is opened).

------------------------------

End of unix/faq Digest part 4 of 7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

Ted Timar - tmatimar@isgtec.com

ISG Technologies Inc., 6509 Airport Road, Mississauga, Ontario, Canada L4V 1S7

------------------------------------------------------------------------

Part1 - Part2 - Part3 - Part4 - Part5 - Part6 - Part7 - MultiPage

Rate this FAQ

[Vote]

Current Top-Rated FAQs

Not finding the answers you need? Submit your question for others to see.

Are you an expert in this area? Share your knowledge and earn expert

points by giving answers or rating people's questions and answers! This

section of FAQS.ORG is not sanctioned in any way by FAQ authors or

maintainers.

Questions strongly related to this FAQ:

\* i\'ma newbie ,can anybody please send me an article about FreeBSD

from instalation, how do... by [-NeWbiE-] (10/16/2003)

\* what is the difference bet. kill-15, kill-9 and other kill commands

in UNIX? by hari (7/14/2003)

\* I am a beginner in UNIX, I have accidentally put a large file in my

workspace and since... by nette (8/26/2003)

\* what is the featuers of UNIX OS by frank (8/15/2003)

\* How do I find the name of the current shell that I am working

on(Whether csh or ksh or... by mulla (10/27/2003)

\* how to create an environment variable by abc (9/14/2003)

\* How to download and upload files on UNIX Server from WIN2K server???

by Arvind (10/28/2003)

\* The Linux kernel does not allow paging out of kernel memory. What

effect does this... by shida (7/20/2003)

\* Using Microport System V release 4, can the Maximum Transmission Unit

be increased to... by Donn (7/21/2003)

Questions somewhat related to this FAQ:

\* I want to know under what condition(s) the following error will be

printed? I have a... by Vidya (7/29/2003)

\* HI ! I am compiling a program where perl is embedded in C on HPUX

64 bit... by Nipun Sharma (9/26/2003)

\* I am relatively new to UNIX and C++ programming, but I want to know

can you (1).... by vhc (6/27/2003)

Other questions awaiting answers:

\* 7054 questions related to other FAQs

\* 1211 general questions

\* 2238 answered questions

---------------------------------------------------------

[ Usenet FAQs | Search | Web FAQs | Documents | RFC Index ]

---------------------------------------------------------

Send corrections/additions to the FAQ Maintainer:

tmatimar@isgtec.com (Ted Timar)

Last Update December 12 2003 @ 00:51 AM

Part1 - Part2 - Part3 - Part4 - Part5 - Part6 - Part7 - MultiPage

------------------------------------------------------------------------

Unix - Frequently Asked Questions (5/7) [Frequent posting]

Not finding the answers

you need?

Submit your question for

others to see.

There are reader questions

on this topic!

Help others by sharing your

knowledge

------------------------------------------------------------------------

Message-ID:

X-Last-Updated: 1996/06/11

From: tmatimar@isgtec.com (Ted Timar)

Newsgroups: comp.unix.questions, comp.unix.shell

Subject: Unix - Frequently Asked Questions (5/7) [Frequent posting]

Date: 01 Dec 2003 09:14:55 GMT

Archive-name: unix-faq/faq/part5

Version: $Id: part5,v 2.9 1996/06/11 13:07:56 tmatimar Exp $

These seven articles contain the answers to some Frequently Asked

Questions often seen in comp.unix.questions and comp.unix.shell.

Please don't ask these questions again, they've been answered plenty

of times already - and please don't flame someone just because they may

not have read this particular posting. Thank you.

This collection of documents is Copyright (c) 1994, Ted Timar, except

Part 6, which is Copyright (c) 1994, Pierre Lewis and Ted Timar.

All rights reserved. Permission to distribute the collection is

hereby granted providing that distribution is electronic, no money

is involved, reasonable attempts are made to use the latest version

and all credits and this copyright notice are maintained.

Other requests for distribution will be considered. All reasonable

requests will be granted.

All information here has been contributed with good intentions, but

none of it is guaranteed either by the contributors or myself to be

accurate. The users of this information take all responsibility for

any damage that may occur.

Many FAQs, including this one, are available on the archive site

rtfm.mit.edu in the directory pub/usenet/news.answers.

The name under which a FAQ is archived appears in the "Archive-Name:"

line at the top of the article. This FAQ is archived as

"unix-faq/faq/part[1-7]".

These articles are divided approximately as follows:

1.\*) General questions.

2.\*) Relatively basic questions, likely to be asked by beginners.

3.\*) Intermediate questions.

4.\*) Advanced questions, likely to be asked by people who thought

they already knew all of the answers.

5.\*) Questions pertaining to the various shells, and the differences.

6.\*) An overview of Unix variants.

7.\*) An comparison of configuration management systems (RCS, SCCS).

This article includes answers to:

5.1) Can shells be classified into categories?

5.2) How do I "include" one shell script from within another

shell script?

5.3) Do all shells have aliases? Is there something else that

can be used?

5.4) How are shell variables assigned?

5.5) How can I tell if I am running an interactive shell?

5.6) What "dot" files do the various shells use?

5.7) I would like to know more about the differences between the

various shells. Is this information available some place?

If you're looking for the answer to, say, question 5.5, and want to skip

everything else, you can search ahead for the regular expression "^5.5)".

While these are all legitimate questions, they seem to crop up in

comp.unix.questions or comp.unix.shell on an annual basis, usually

followed by plenty of replies (only some of which are correct) and then

a period of griping about how the same questions keep coming up. You

may also like to read the monthly article "Answers to Frequently Asked

Questions" in the newsgroup "news.announce.newusers", which will tell

you what "UNIX" stands for.

With the variety of Unix systems in the world, it's hard to guarantee

that these answers will work everywhere. Read your local manual pages

before trying anything suggested here. If you have suggestions or

corrections for any of these answers, please send them to to

tmatimar@isgtec.com.

---------------------------------------------------------------

Subject: Can shells be classified into categories?

>From: wicks@dcdmjw.fnal.gov (Matthew Wicks)

Date: Wed, 7 Oct 92 14:28:18 -0500

5.1) Can shells be classified into categories?

In general there are two main class of shells. The first class

are those shells derived from the Bourne shell which includes sh,

ksh, bash, and zsh. The second class are those shells derived

from C shell and include csh and tcsh. In addition there is rc

which most people consider to be in a "class by itself" although

some people might argue that rc belongs in the Bourne shell class.

With the classification above, using care, it is possible to

write scripts that will work for all the shells from the Bourne

shell category, and write other scripts that will work for all of

the shells from the C shell category.

---------------------------------------------------------------

Subject: How do I "include" one shell script from within another shell script?

>From: wicks@dcdmjw.fnal.gov (Matthew Wicks)

Date: Wed, 7 Oct 92 14:28:18 -0500

5.2) How do I "include" one shell script from within another shell script?

All of the shells from the Bourne shell category (including rc)

use the "." command. All of the shells from the C shell category

use "source".

---------------------------------------------------------------

Subject: Do all shells have aliases? Is there something else that can be used?

>From: wicks@dcdmjw.fnal.gov (Matthew Wicks)

Date: Wed, 7 Oct 92 14:28:18 -0500

5.3) Do all shells have aliases? Is there something else that can be used?

All of the major shells other than sh have aliases, but they

don't all work the same way. For example, some don't accept

arguments.

Although not strictly equivalent, shell functions (which exist in

most shells from the Bourne shell category) have almost the same

functionality of aliases. Shell functions can do things that

aliases can't do. Shell functions did not exist in bourne shells

derived from Version 7 Unix, which includes System III and BSD 4.2.

BSD 4.3 and System V shells do support shell functions.

Use unalias to remove aliases and unset to remove functions.

---------------------------------------------------------------

Subject: How are shell variables assigned?

>From: wicks@dcdmjw.fnal.gov (Matthew Wicks)

Date: Wed, 7 Oct 92 14:28:18 -0500

5.4) How are shell variables assigned?

The shells from the C shell category use "set variable=value" for

variables local to the shell and "setenv variable value" for

environment variables. To get rid of variables in these shells

use unset and unsetenv. The shells from the Bourne shell

category use "variable=value" and may require an "export

VARIABLE\_NAME" to place the variable into the environment. To

get rid of the variables use unset.

---------------------------------------------------------------

Subject: How can I tell if I am running an interactive shell?

>From: wicks@dcdmjw.fnal.gov (Matthew Wicks)

>From: dws@ssec.wisc.edu (DaviD W. Sanderson)

Date: Fri, 23 Oct 92 11:59:19 -0600

5.5) How can I tell if I am running an interactive shell?

In the C shell category, look for the variable $prompt.

In the Bourne shell category, you can look for the variable $PS1,

however, it is better to check the variable $-. If $- contains

an 'i', the shell is interactive. Test like so:

case $- in

\*i\*) # do things for interactive shell

;;

\*) # do things for non-interactive shell

;;

esac

---------------------------------------------------------------

Subject: What "dot" files do the various shells use?

>From: wicks@dcdmjw.fnal.gov (Matthew Wicks)

>From: tmb@idiap.ch (Thomas M. Breuel)

Date: Wed, 28 Oct 92 03:30:36 +0100

5.6) What "dot" files do the various shells use?

Although this may not be a complete listing, this provides the

majority of information.

csh

Some versions have system-wide .cshrc and .login files. Every

version puts them in different places.

Start-up (in this order):

.cshrc - always; unless the -f option is used.

.login - login shells.

Upon termination:

.logout - login shells.

Others:

.history - saves the history (based on $savehist).

tcsh

Start-up (in this order):

/etc/csh.cshrc - always.

/etc/csh.login - login shells.

.tcshrc - always.

.cshrc - if no .tcshrc was present.

.login - login shells

Upon termination:

.logout - login shells.

Others:

.history - saves the history (based on $savehist).

.cshdirs - saves the directory stack.

sh

Start-up (in this order):

/etc/profile - login shells.

.profile - login shells.

Upon termination:

any command (or script) specified using the command:

trap "command" 0

ksh

Start-up (in this order):

/etc/profile - login shells.

.profile - login shells; unless the -p option is used.

$ENV - always, if it is set; unless the -p option is used.

/etc/suid\_profile - when the -p option is used.

Upon termination:

any command (or script) specified using the command:

trap "command" 0

bash

Start-up (in this order):

/etc/profile - login shells.

.bash\_profile - login shells.

.profile - login if no .bash\_profile is present.

.bashrc - interactive non-login shells.

$ENV - always, if it is set.

Upon termination:

.bash\_logout - login shells.

Others:

.inputrc - Readline initialization.

zsh

Start-up (in this order):

.zshenv - always, unless -f is specified.

.zprofile - login shells.

.zshrc - interactive shells, unless -f is specified.

.zlogin - login shells.

Upon termination:

.zlogout - login shells.

rc

Start-up:

.rcrc - login shells

---------------------------------------------------------------

Subject: I would like to know more about the differences ... ?

>From: wicks@dcdmjw.fnal.gov (Matthew Wicks)

Date: Wed, 7 Oct 92 14:28:18 -0500

5.7) I would like to know more about the differences between the

various shells. Is this information available some place?

A very detailed comparison of sh, csh, tcsh, ksh, bash, zsh, and

rc is available via anon. ftp in several places:

ftp.uwp.edu (204.95.162.190):pub/vi/docs/shell-100.BetaA.Z

utsun.s.u-tokyo.ac.jp:misc/vi-archive/docs/shell-100.BetaA.Z

This file compares the flags, the programming syntax,

input/output redirection, and parameters/shell environment

variables. It doesn't discuss what dot files are used and the

inheritance for environment variables and functions.

------------------------------

End of unix/faq Digest part 5 of 7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

Ted Timar - tmatimar@isgtec.com

ISG Technologies Inc., 6509 Airport Road, Mississauga, Ontario, Canada L4V 1S7

------------------------------------------------------------------------

Part1 - Part2 - Part3 - Part4 - Part5 - Part6 - Part7 - MultiPage

Rate this FAQ

[Vote]

Current Top-Rated FAQs

Not finding the answers you need? Submit your question for others to see.

Are you an expert in this area? Share your knowledge and earn expert

points by giving answers or rating people's questions and answers! This

section of FAQS.ORG is not sanctioned in any way by FAQ authors or

maintainers.

Questions strongly related to this FAQ:

\* i\'ma newbie ,can anybody please send me an article about FreeBSD

from instalation, how do... by [-NeWbiE-] (10/16/2003)

\* what is the difference bet. kill-15, kill-9 and other kill commands

in UNIX? by hari (7/14/2003)

\* I am a beginner in UNIX, I have accidentally put a large file in my

workspace and since... by nette (8/26/2003)

\* what is the featuers of UNIX OS by frank (8/15/2003)

\* How do I find the name of the current shell that I am working

on(Whether csh or ksh or... by mulla (10/27/2003)

\* how to create an environment variable by abc (9/14/2003)

\* How to download and upload files on UNIX Server from WIN2K server???

by Arvind (10/28/2003)

\* The Linux kernel does not allow paging out of kernel memory. What

effect does this... by shida (7/20/2003)

\* Using Microport System V release 4, can the Maximum Transmission Unit

be increased to... by Donn (7/21/2003)

Questions somewhat related to this FAQ:

\* I want to know under what condition(s) the following error will be

printed? I have a... by Vidya (7/29/2003)

\* HI ! I am compiling a program where perl is embedded in C on HPUX

64 bit... by Nipun Sharma (9/26/2003)

\* I am relatively new to UNIX and C++ programming, but I want to know

can you (1).... by vhc (6/27/2003)

Other questions awaiting answers:

\* 7054 questions related to other FAQs

\* 1211 general questions

\* 2238 answered questions

---------------------------------------------------------

[ Usenet FAQs | Search | Web FAQs | Documents | RFC Index ]

---------------------------------------------------------

Send corrections/additions to the FAQ Maintainer:

tmatimar@isgtec.com (Ted Timar)

Last Update December 12 2003 @ 00:51 AM

Part1 - Part2 - Part3 - Part4 - Part5 - Part6 - Part7 - MultiPage

------------------------------------------------------------------------

Unix - Frequently Asked Questions (6/7) [Frequent posting]

Not finding the answers

you need?

Submit your question for

others to see.

There are reader questions

on this topic!

Help others by sharing your

knowledge

------------------------------------------------------------------------

Message-ID:

X-Last-Updated: 1996/06/11

From: tmatimar@isgtec.com (Ted Timar)

Newsgroups: comp.unix.questions, comp.unix.shell

Subject: Unix - Frequently Asked Questions (6/7) [Frequent posting]

Date: 01 Dec 2003 09:14:55 GMT

Archive-name: unix-faq/faq/part6

Version: $Id: part6,v 2.9 1996/06/11 13:07:56 tmatimar Exp $

These seven articles contain the answers to some Frequently Asked

Questions often seen in comp.unix.questions and comp.unix.shell.

Please don't ask these questions again, they've been answered plenty

of times already - and please don't flame someone just because they may

not have read this particular posting. Thank you.

This collection of documents is Copyright (c) 1994, Ted Timar, except

Part 6, which is Copyright (c) 1994, Pierre Lewis and Ted Timar.

All rights reserved. Permission to distribute the collection is

hereby granted providing that distribution is electronic, no money

is involved, reasonable attempts are made to use the latest version

and all credits and this copyright notice are maintained.

Other requests for distribution will be considered. All reasonable

requests will be granted.

All information here has been contributed with good intentions, but

none of it is guaranteed either by the contributors or myself to be

accurate. The users of this information take all responsibility for

any damage that may occur.

Many FAQs, including this one, are available on the archive site

rtfm.mit.edu in the directory pub/usenet/news.answers.

The name under which a FAQ is archived appears in the "Archive-Name:"

line at the top of the article. This FAQ is archived as

"unix-faq/faq/part[1-7]".

These articles are divided approximately as follows:

1.\*) General questions.

2.\*) Relatively basic questions, likely to be asked by beginners.

3.\*) Intermediate questions.

4.\*) Advanced questions, likely to be asked by people who thought

they already knew all of the answers.

5.\*) Questions pertaining to the various shells, and the differences.

6.\*) An overview of Unix variants.

7.\*) An comparison of configuration management systems (RCS, SCCS).

This article includes answers to:

6.1) Disclaimer, introduction and acknowledgements.

6.2) A very brief look at Unix history.

6.3) Main Unix flavors.

6.4) Unix Standards.

6.5) Identifying your Unix flavor.

6.6) Brief notes on some well-known (commercial/PD) Unices.

6.7) Real-time Unices.

6.8) Unix glossary.

If you're looking for the answer to, say, question 6.5, and want to skip

everything else, you can search ahead for the regular expression "^6.5)".

While these are all legitimate questions, they seem to crop up in

comp.unix.questions or comp.unix.shell on an annual basis, usually

followed by plenty of replies (only some of which are correct) and then

a period of griping about how the same questions keep coming up. You

may also like to read the monthly article "Answers to Frequently Asked

Questions" in the newsgroup "news.announce.newusers", which will tell

you what "UNIX" stands for.

With the variety of Unix systems in the world, it's hard to guarantee

that these answers will work everywhere. Read your local manual pages

before trying anything suggested here. If you have suggestions or

corrections for any of these answers, please send them to to

tmatimar@isgtec.com.

---------------------------------------------------------------

Subject: Disclaimer, introduction and acknowledgements.

>From: "Pierre (P.) Lewis" <lew@bnr.ca>

Date: Tue Aug 15 15:14:00 EDT 1995

X-Version: 2.9

6.1) Disclaimer, introduction and acknowledgements.

NOTE TO READERS: I would like to update this FAQ with WWW pointers

for the various Unices I mention. Don't hesitate to send them along,

I'll eventually get around to updating this part. Email: lew@bnr.ca

The following is offered with no guarantee as to accuracy or

completeness. I have done what I can in the time available,

often with conflicting information, and it still is very much work

in progress. I hope to keep improving this summary. Comments and

corrections welcome: lew@bnr.ca.

First a short definition. By Unix we mean an operating system

typically written in C, with a hierarchical file system,

integration of file and device I/O, whose system call interface

includes services such as fork(), pipe(), and whose user

interface includes tools such as cc, troff, grep, awk, and a

choice of shell. Note that UNIX was a registered trademark of USL

(AT&T), now of X/Open, but will be used here in its generic sense.

Most Unices (the more common plural form) are derived more or

less directly from AT&T (now Novell) code (some code from the first C

version is presumably still left in most), but there are also clones

(i.e. Unix-compatible systems with no AT&T code).

In addition, there are also Unix-like environments (e.g. VOS)

sitting on top of other OSs, and OSs inspired from Unix (yes,

even DOS!). These are not covered here. Little on real-time

Unices yet (although more is planned).

Unix comes in an incredible variety of flavors. This is to a

large extent due to availability of sources and the ease of

porting and modifying Unix. Typically, a vendor of Unix will

start with one basic flavor (see below), take ideas/code from the

other major flavor, add and change many things, etc. This

results in yet another new Unix flavor. Today, there are

literally hundreds of Unices available, the closest thing to

standard Unix being (by definition) System V.

This answer was put together mostly from information on the net

and email. Some specific sources are also mentioned in the

appropriate sections.

Acknowledgements: (in addition to references): pat@bnr.ca,

guy@auspex.com, pen@lysator.liu.se, mikes@ingres.com,

mjd@saul.cis.upenn.edu, root%candle.uucp@ls.com, ee@atbull.bull.co.at,

Aaron\_Dailey@stortek.com, ralph@dci.pinetree.org, sbdah@mcshh.hanse.de,

macmach@andrew.cmu.edu, jwa@alw.nih.gov [4.4BSD], roeber@axpvms.cern.ch,

bob@pta.pyramid.com.au, bad@flatlin.ka.sub.org, m5@vail.tivoli.com,

dan@fch.wimsey.bc.ca, jlbrand@uswnvg.com, jpazer@usl.com,

ym@satelnet.org, merritt@gendev.slc.paramax.com, quinlan@yggdrasil.com,

steve@rudolph.ssd.csd.harris.com, bud@heinous.isca.uiowa.edu,

pcu@umich.edu, quinlan@yggdrasil.com, Dan\_Menchaca@quickmail.apple.com,

D.Lamptey@sheffield.ac.uk, derekn@vw.ece.cmu.edu, gordon@PowerOpen.org,

romain@pyramid.com, rzm@dain.oso.chalmers.se, chen@adi.com,

tbm@tci002.uibk.ac.at, sllewis@nando.net, edwin@modcomp.demon.co.uk,

many that I forgot, and all the other folks whose posts I read. Many

thanks!

---------------------------------------------------------------

Subject: A very brief look at Unix history.

>From: "Pierre (P.) Lewis"

Date: Mon May 30 15:44:28 EDT 1994

X-Version: 2.6

6.2) A very brief look at Unix history.

Unix history goes back to 1969 and the famous "little-used PDP-7

in a corner" on which Ken Thompson, Dennis Ritchie (the R in K&R)

and others started work on what was to become Unix. The name

"Unix" was intended as a pun on Multics (and was written "Unics"

at first -- UNiplexed Information and Computing System).

For the first 10 years, Unix development was essentially confined

to Bell Labs. These initial versions were labeled "Version n" or

"Nth Edition" (of the manuals), and were for DEC's PDP-11 (16

bits) and later VAXen (32 bits). Some significant versions

include:

V1 (1971): 1st Unix version, in assembler on a PDP-11/20.

Included file system, fork(), roff, ed. Was used as a text

processing tool for preparation of patents. Pipe() appeared

first in V2!

V4 (1973): Rewritten in C, which is probably the most

significant event in this OS's history: it means Unix can be

ported to a new hardware in months, and changes are easy. The

C language was originally designed for the Unix operating

system, and hence there is a strong synergy between C and Unix.

V6 (1975): First version of Unix widely available outside

Bell Labs (esp. in universities). This was also the start of

Unix diversity and popularity. 1.xBSD (PDP-11) was derived

from this version. J. Lions published "A commentary on the

Unix Operating System" based on V6.

V7 (1979): For many, this is the "last true Unix", an

"improvement over all preceding and following Unices"

[Bourne]. It included full K&R C, uucp, Bourne shell. V7 was

ported to the VAX as 32V. The V7 kernel was a mere 40

Kbytes!

Here (for reference) are the system calls of V7:

\_exit, access, acct, alarm, brk, chdir, chmod, chown,

chroot, close, creat, dup, dup2, exec\*, exit, fork, fstat,

ftime, getegid, geteuid, getgid, getpid, getuid, gtty,

indir, ioctl, kill, link, lock, lseek, mknod, mount,

mpxcall, nice, open, pause, phys, pipe, pkoff, pkon,

profil, ptrace, read, sbrk, setgid, setuid, signal, stat,

stime, stty, sync, tell, time, times, umask, umount,

unlink, utime, wait, write.

These Vn versions were developed by the Computer Research Group

(CRG) of Bell Labs. Another group, the Unix System Group (USG),

was responsible for support. A third group at Bell Labs was also

involved in Unix development, the Programmer's WorkBench (PWB),

to which we owe, for example, sccs, named pipes and other

important ideas. Both groups were merged into Unix System

Development Lab in 1983.

Another variant of Unix was CB Unix (Columbus Unix) from the Columbus

branch of Bell Labs, responsible of Operations Support Systems. Its

main contribution was parts of SV IPC.

Work on Unix continued at Bell Labs in the 1980s. The V series

was further developed by the CRG (Stroustrup mentions V10 in the

2nd edition of his book on C++), but we don't seem to hear much

about this otherwise. The company now responsible for Unix

(System V) is called Unix System Laboratories (USL) and is

majority-owned by AT&T. Novell has bought USL (early 93)!

Novell has given rights to the "UNIX" trademark to X/Open (late 93).

But much happened to Unix outside AT&T, especially at Berkeley

(where the other major flavor comes from). Vendors (esp. of

workstations) also contributed much (e.g. Sun's NFS).

The book "Life with Unix" by Don Libes and Sandy Ressler is

fascinating reading for anyone interested in Unix, and covers a

lot of the history, interactions, etc.. Much in the present

section is summarized from this book.

---------------------------------------------------------------

Subject: Main Unix flavors.

>From: "Pierre (P.) Lewis"

Date: Mon Jan 9 16:59:14 EST 1995

X-Version: 2.7

6.3) Main Unix flavors.

The following is very much an early '90s view.

Until recently, there were basically two main flavors of Unix:

System V (five) from AT&T, and the Berkeley Software Distribution

(BSD). SVR4 is essentially a merge of these two flavors. End

'91, OSF/1 from the Open Software Foundation was released (as a

direct competitor to System V) and may (future will tell) change

this picture.

The following lists the main releases and features of System V,

BSD and OSF/1.

System V from AT&T. Typical of Intel hardware. Most often

ported Unix, typically with BSD enhancements (csh, job

control, termcap, curses, vi, symbolic links). System V

evolution is now overseen by Unix International (UI). UI

members include AT&T, Sun, ....

Newsgroup: comp.unix.sysv[23]86. Main releases:

- System III (1982): first commercial Unix from AT&T

- FIFOs (named pipes) (later?)

- System V (1983):

- IPC package (shm, msg, sem)

- SVR2 (1984):

- shell functions (sh)

- SVID (System V Interface Definition)

- SVR3 (1986) for ? platforms:

- STREAMS (inspired by V8), poll(), TLI (network software)

- RFS

- shared libs

- SVID 2

- demand paging (if hardware supports)

- SVR3.2:

- merge with Xenix (Intel 80386)

- networking

- SVR4 (1988), mainstream of Unix implementations, merge of

System V, BSD, and SunOS.

- From SVR3: sysadmin, terminal I/F, printer (from BSD?),

RFS, STREAMS, uucp

- From BSD: FFS, TCP/IP, sockets, select(), csh

- From SunOS: NFS, OpenLook GUI, X11/NeWS, virtual memory

subsystem with memory-mapped files, shared libraries

(!= SVR3 ones?)

- ksh

- ANSI C

- Internationalization (8-bit clean)

- ABI (Application Binary Interface -- routines instead of traps)

- POSIX, X/Open, SVID3

- SVR4.1

- async I/O (from SunOS?)

- SVR4.2 (based on SVR4.1ES)

- Veritas FS, ACLs

- Dynamically loadable kernel modules

- Future:

- SVR4 MP (multiprocessor)

- Use of Chorus microkernel?

Berkeley Software Distribution (BSD). Typical of VAXen, RISCs,

many workstations. More dynamic, research versions now than

System V. BSD is responsible for much of the popularity of

Unix. Most enhancements to Unix started here. The group

responsible at UCB (University of California at Berkeley) is

the Computer System Research Group (CSRG). They closed down

in 1992. Newsgroup: comp.unix.bsd. Main releases:

(much reorganized wrt dates and releases, hope it's converging)

- 2.xBSD (1978) for PDP-11, still of significance? (2.11BSD

was released in 1992!).

- csh

- 3BSD (1978):

- virtual memory

- 4.?BSD:

- termcap, curses

- vi

- 4.0BSD (1980):

- 4.1BSD (?): base of later AT&T CRG versions

- job control

- automatic kernel config

- vfork()

- 4.2BSD (1983):

- TCP/IP, sockets, ethernet

- UFS: long file names, symbolic links

- new reliable signals (4.1 reliable signals now in SVR3)

- select()

- 4.3BSD (1986) for VAX, ?:

- 4.3 Tahoe (1988): 4.3BSD with sources, support for Tahoe

(32-bit supermini)

- Fat FFS

- New TCP algorithms

- 4.3 Reno (1990) for VAX, Tahoe, HP 9000/300:

- most of P1003.1

- NFS (from Sun)

- MFS (memory file system)

- OSI: TP4, CLNP, ISODE's FTAM, VT and X.500; SLIP

- Kerberos

- Net1 (?) and Net2 (June 1991) tapes: that portion of BSD which

requires no USL copyright

- 4.4BSD (alpha June 1992) for HP 9000/300, Sparc, 386, DEC, others;

neither VAX nor Tahoe; two versions, lite (~Net2 contents plus,

fixes and new architectures) and encumbered (everything, requires

USL license):

- new virtual memory system (VMS) based on Mach 2.5

- virtual filesystem interface, log-structured filesystem, size

of local filesystem up to 2^63, NFS (freely redistributable,

works with Sun's, over UDP or TCP)

- ISO/OSI networking support (based on ISODE): TP4/CLNP/802.3 and

TP0/CONS/X.25, session and above in user space; FTAM, VT, X.500.

- most of POSIX.1 (esp. new terminal driver a la SV), much of

POSIX.2, improved job control; ANSI C headers

- Kerberos integrated with much of the system (incl. NFS)

- TCP/IP enhancements (incl. header prediction, SLIP)

- important kernel changes (new system call convention, ...)

- other improvements: FIFOs, byte-range file locking

Official 4.4BSD release was expected within 6 months of above.

The Open Software Foundation (OSF) released its Unix called OSF/1

end of 1991. Still requires an SVR2 license.

Compatible/compliant with SVID 2 (and 3 coming), POSIX,

X/Open, etc.. OSF members include Apollo, Dec, HP, IBM, ....

- OSF/1 (1991):

- based on Mach 2.5 kernel

- symmetric multiprocessing, parallelized kernel, threads

- logical volumes, disk mirroring, UFS (native), S5 FS, NFS

- enhanced security (B1 with some B2, B3; or C2), 4.3BSD admin

- STREAMS, TLI/XTI, sockets

- shared libs, dynamic loader (incl. kernel)

- Motif GUI

- Release 1.3 (Jun 94)

- Based on MACH 3.0 Micro-kernel

- Conformant with current draft of Specification 1170

(considered for standardization in X/Open's Fast Track process)

- Data Capture I/F, Common Data Link I/F,

- ISO 10646 and 64-bit support.

- OSF/1 MK (mikrokernel) based on Mach 3.0

This list of major flavors should probably also include Xenix

(Microsoft) which has been the basis for many ports. Derived from V7,

S III and finally System V, it is similar externally but significantly

changed internally (performance-tuned for micros).

Two very good books describe the internals of the two main flavors.

These are:

- System V: "Design of the Unix Operating System", M.J. Bach.

- BSD: "Design and Implementation of the 4.3BSD Unix Operating System",

Leffler, McKusick, Karels, Quaterman.

For a good introduction to OSF/1 (not quite as technical as the

previous two), see: "Guide to OSF/1, A Technical Synopsis",

published by O'Reilly. On SunOS, "Virtual Memory Architecture in

SunOS" and "Shared Libraries in SunOS" in Summer 1989 USENIX

Proceedings.

A good set of articles on where Unix is going is "Unix Variants"

in the Apr 92 issue of Unix Review. Other good sources of

information include the bsd-faq file, and many of the newsgroups

mentioned in the text.

---------------------------------------------------------------

Subject: Main Players and Unix Standards.

>From: "Pierre (P.) Lewis"

Date: Mon Jan 21 16:59:14 EST 1995

X-Version: 2.8

6.4) Main Players and Unix Standards.

The more important players in the Unix scene currently (early '95)

are (corrections most welcome, these are new bytes):

- Novell who bought USL (early 93) and now has the source code.

- X/Open who has the branding rights to "UNIX" trademark.

- OSF, both as developer of OSF/1 and Motif, and as organization

overseeing COSE (OSF's new working model). OSF was reorganized in

1994 (and Sun joined), relationship with X/Open has been formalized.

- IEEE with POSIX, LAN standards.

- PowerOpen [IBM, Apple, Motorola, Bull, others] promoting the PowerPC.

Do not confuse with graphical environment of same name.

The following briefly describes the more important standards

relevant to Unix.

- IEEE:

- 802.x (LAN) standards (LLC, ethernet, token ring, token bus)

- POSIX (ISO 9945?): Portable Operating System I/F (Unix, VMS

and OS/2!) (only ? have been finalized at this point)

- 1003.1: library procedures (mostly system calls) -- roughly V7

except for signals and terminal I/F (1990)

- 1003.2: shell and utilities

- 1003.3: test methods and conformance

- 1003.4: real-time: binary semaphores, process memory

locking, memory-mapped files, shared memory,

priority scheduling, real-time signals, clocks and

timers, IPC message passing, synchronized I/O,

asynchronous I/O, real-time files

- 1003.5: Ada language bindings

- 1003.6: security

- 1003.7: system admin (incl. printing)

- 1003.8: transparent file access

- 1003.9: FORTRAN language bindings

- 1003.10: super computing

- 1003.12: protocol-independent I/Fs

- 1003.13: real-time profiles

- 1003.15: supercomputing batch I/Fs

- 1003.16: C-language bindings (?)

- 1003.17: directory services

- 1003.18: POSIX standardized profile

- 1003.19: FORTRAN 90 language bindings

- X/Open (consortium of vendors, founded 1984):

- X/Open Portability Guides (XPGn):

- XPG2 (1987), strong SV influence

Vol 1: commands and utilities

Vol 2: system calls and libraries

Vol 3: terminal I/F (curses, termio), IPC (SV),

internationalization

Vol 4: programming languages (C, COBOL!)

Vol 5: data management (ISAM, SQL)

- XPG3 (1989) adds: X11 API

- XPG4 (1992) adds: XTI? 22 components

- XOM series of interfaces:

- XOM (X/Open Object Management) generic I/F mechanisms for

following

- XDS (X/Open Directory Service)

- XMH (X/Open Mail ??)

- XMP (X/Open Management Protocols) -- not Bull's CM API?

- X/Open now has the rights to the "UNIX" trademark (late 93);

- "Spec 1170"

- This specification is being prepared describing a common API

to which vendors wanting to use the name "UNIX" will have to

comply (when test suites are available). Merge of SVID,

OSF's AES and other stuff.

- AT&T

(is this still relevant in 1994? Who is now responsible for SVID,

TLI, APLI?)

- System V Interface Definition (SVID)

- SVID1 (1985, SVR2)

Vol 1: system calls and libraries (similar to XPG2.1)

- SVID2 (1986, SVR3)

Vol 1: system calls and libraries (base, kernel extensions)

Vol 2: commands and utilities (base, advanced, admin, software

development), terminal I/F

Vol 3: terminal I/F (again), STREAMS and TLI, RFS

- SVID3 (19??, SVR4) adds

Vol 4: ?? &c

- APIs

- Transport Library Interface (TLI)

- ACSE/Presentation Library Interface (APLI)

- COSE (COmmon Open Software Environment) [IBM, HP, SunSoft, others]:

objective is to bring different Unix platforms closer together.

Initiatives in the following areas:

- desktop environments

- application API (aka Spec 1170 -- a single programming i/f) --

probably the more important achievement at this point: eliminates

differences between SCO, AIX, Solaris, HP-UX, UnixWare.

- distributed computing services (OSF's DCE and SunSoft's ONC)

- object technologies (OMG's CORBA)

- graphics

- multimedia

- systems management

- PowerOpen Environment (POE) promoted by the PowerOpen association

(POA). A standard for Unix-like OSs running on PowerPC chip. Defines:

- an API (application programming i/f, derived from AIX, conforms to

POSIX, XPG4, Motif, &c) and

- an ABI (application binary i/f), a distinguishing factor from other

standards such as POSIX, XPG4, &c.. Any POE-compliant system will

be able to run all POE software.

Key features:

- based on the PowerPC architecture

- hardware bus independence

- system implementations can range from laptops to supercomputers

- requires a multi-user, multi-tasking operating system

- networking support

- X windows extension, Motif

- conformance tested and certified by an independent party (POA)

AIX 4.1.1 will be PowerOpen compliant. MacOS isn't and won't be.

[above adapted from the powerpc-faq from comp.sys.powerpc]

IBM is involved in both COSE and POE. How will the two interact?

---------------------------------------------------------------

Subject: Identifying your Unix flavor.

>From: "Pierre (P.) Lewis"

Date: Mon May 30 15:44:28 EDT 1994

X-Version: 2.6

6.5) Identifying your Unix flavor.

This section lists a number of things you can look at in

attempting to identify the base flavor of your Unix. Given the

significant exchange of code and ideas between the various

flavors and the many changes made by vendors, any statement such

as "this Unix is an SVR2" is at best a statistical statement

(except for some SVRn ports). Also many Unices offer most of

both worlds (either mixed as in SunOS or strictly separated as in

Apollo?). So this section is perhaps not very useful...

The list of features in previous sections can also help. For

example, if a system has a poll(2) but no select(2), it is highly

probable that it is derived from SVR3. Also the name of the OS

can provide a clue, as well as the logon message (e.g. SGI's

"IRIX SVR3.3.2") or the output of "uname -a" command. Available

commands can also provide hints but this is probably less

reliable than kernel features. For example, the type of terminal

initialization (inittab or ttys) is a more reliable indicator

than the print subsystem.

Feature Typical in SVRx Typical in xBSD

kernel name /unix /vmunix

terminal init /etc/inittab /etc/ttys (only getty to 4.3)

boot init /etc/rc.d directories /etc/rc.\* files

mounted FSs /etc/mnttab /etc/mtab

usual shell sh, ksh csh, #! hack

native FS S5 (blk: 512-2K) UFS (blk: 4K-8K)

file names <= 14 bytes file names < 255 bytes

groups need newgrp(1) automatic membership

SVR4: multiple groups

print subsystem lp, lpstat, cancel lpr, lpq, lprm (lpd daemon) ??

terminal control termio, terminfo, termios (sgtty before 4.3reno)

SVR4: termios (POSIX) termcap

job control >= SVR4 yes

ps command ps -ef ps -aux

multiple wait poll select

string fcns memset, memcpy bzero, bcopy

process mapping /proc (SVR4)

As we move to the late '90s, this is probably less and less relevant.

---------------------------------------------------------------

Subject: Brief notes on some well-known (commercial/PD) Unices.

>From: "Pierre (P.) Lewis"

Date: Tue Aug 15 15:14:00 EDT 1995

X-Version: 2.9

6.6) Brief notes on some well-known (commercial/PD) Unices.

(I am not at all satisfied with this section, unfortunately I

have neither the time nor the documents to make it much better

(wrt contents). Should only list Unices known by a reasonably

wide audience. Small and non-US Unices welcome, e.g. Eurix. In

need of reformatting)

This section lists (in alphabetical order) some of the better

known Unices along with a brief description of their nature.

Unfortunately, it's out-of-date almost by definition...

(sorted alpha, ignoring numbers and other chars)

AIX: IBM's Unix, based on SVR2 (later up to SVR3.2?) with varying

degrees of BSD extensions, for various hardwares. Proprietary

system admin (SMIT). Both 850 and Latin-1 CPs. Quite

different from most Unices and among themselves.

Newsgroup: comp.unix.aix.

- 1.x (for 386 PS/2)

- 2.x (for PC RTs)

- 3.x (for RS/6000), paging kernel, logical volume manager, i18n;

3.2 adds TLI/STREAMS. SV-based with many enhancements.

4.1 is latest (includes support for PowerPC?)

- AIX/ESA, runs native on S/370 and S/390 mainframes, based on OSF/1.

AIX was to have been base for OSF/1 until Mach was chosen instead.

I hope this subsection is converging :-)

AOS (IBM): 4.3BSD port to IBM PC RT (for educational institutes).

Don't confuse with DG's proprietary OS of same name.

Arix: SV

A3000UX (Commodore): 68030-based SVR4 Unix (?) for the Amiga.

A/UX (Apple): SV with Berkeley enhancements, NFS, Mac GUI. System 6

(later System 7) runs as guest of A/UX (opposite of MachTen).

Newsgroup: comp.unix.aux.

- 2.0: SVR2 with 4.2BSD, system 6 Mac applications.

- 3.0 (1992): SVR2.2 with 4.3BSD and SVR3/4 extensions; X11R4,

MacX, TCP/IP, NFS, NIS, RPC/XDR, various shells, UFS or S5FS.

System 7 applications.

- 4.0 will have/be OSF/1. But I hear Apple has decided to drop

A/UX (will go for AIX now that they're together with IBM on

the PPC)

3B1 (680x0): SV-based, done by Convergent for AT&T.

Newsgroup: comp.sys.3b1.

BNR/2: stands for BSD Net/2 Release? Includes NetBSD/1, FreeBSD.

BOS for Bull's DPX/2 (680x0)

- V1 (1990): SVR3 with BSD extensions (FFS, select, sockets),

symmetric MP, X11R3

- V2 (1991): adds job control, disk mirroring, C2 security,

DCE extensions

- There's also BOS/X, and AIX-compatible Unix for Bull's PPC

workstations. How it relates to above two is unknown.

386BSD: Jolitz's port of Net/2 software. Posix, 32-bit, still in alpha

(now version 0.1).

BSD/386 (80386): from BSDI, with source (augmented Net2 software)

Newsgroup: comp.unix.bsd.

Chorus/MiXV: Unix SVR3.2 (SVR4) over Chorus nucleus, ABI/BCS.

Coherent (Mark Williams Company): For 80286. Unix clone compatible with

V7, some SVR2 (IPC). V4.0 is 32-bit. Newsgroup: comp.os.coherent.

Mark Williams closed down early '95.

Consensys: SVR4.2

CTIX: SV-based, from Convergent

D-NIX: SV

DC/OSx (Pyramid): SVR4. Newsgroup: comp.sys.pyramid.

DELL UNIX [DELL Computer Corp.]: SVR4

DomainIX: see DomainOS below.

DomainOS (Apollo, now HP): proprietary OS; layered on top is BSD4.3 and

SVR3 (a process can use either, neither or both). Development now

stopped, some features now in OSF/1 (and NT). Now at SR10.4.

Name for SR9.\* was DomainIX. Newsgroup: comp.sys.apollo.

DVIX (NT's DVS): SVR2

DYNIX (Sequent): 4.2BSD-based

DYNIX/PTX: SVR3-based

EP/IX (Control Data Corp.): for MIPS 2000/3000/6000/4000; based on

RISC/OS 4 and 5, POSIX-ABI-compliant. SVR3, SVR4 and BSD modes.

Esix (80386): pure SVR4, X11, OpenLook (NeWS), Xview

Eurix (80?86): SVR3.2 (Germany)

FreeBSD: 386bsd 0.1 with the patchkit applied, and many updated

utilities.

FTX: Stratus fault-tolerant OS (68K or i860-i960 hardware)

Generics UNIX (80386): SVR4.03 (Germany)

GNU Hurd (?): vaporware from the Free Software Foundation (FSF):

Unix emulator over Mach 3.0 kernel. Many GNU tools are very

popular (emacs) and used in the PD Unices.

HELIOS (Perihelion Software): for INMOS transputer and many other

platforms.

HP-UX (HP): old from S III (SVRx), now SVR2 (4.2BSD?) with SV utilities

(they have trouble making up their minds).

- 6.5: SVR2

- 7.0: SVR3.2, symlinks

- 7.5

- 8.0: BSD based? for HP-9000 CISC (300/400) and RISC (800/700),

shared libs

- 9.0: includes DCE

Interactive SVR3.2 (80x86): pure SVR3. Interactive has been bought

by Sun; will their system survive Solaris?

Idris: first Unix clone by Whitesmith. A small Unix? For INMOS

transputer and others?.

IRIX (SGI): Version 4: SVR3.2, much BSD. Version 5.x (current is 5.2)

is based on SVR4. Newsgroup: comp.sys.sgi.

Linux (386/486/586): Unix under GPL (not from FSF, though). Available

with sources. POSIX compliant w/ SysV and BSD extensions. Being

ported to Alpha/AXP and PowerPC (ports for 680x0 Amigas and Ataris

already exist; a port is also being done to the MIPS/4000).

Newsgroup: comp.os.linux.{admin,announce,development,help,misc}.

MacBSD, ?: works on Mac II (directly on H/W).

MachTen, Tenon Intersystems: runs as a guest under MacOS; 4.3BSD

environment with TCP, NFS. Scaled down version: MachTen Personal.

MacMach (Mac II): 4.3BSD over Mach 3.0 microkernel, X11, Motif, GNU

software, sources, experimental System 7 as Mach task. Complete

with all sources (need Unix license).

Mach386: from Mt Xinu. Based on Mach 2.5, with 4.3BSD-Tahoe

enhancements. Also 2.6 MSD (Mach Source Distribution).

Microport (80x86): pure SVR4, X11, OpenLook GUI

Minix (80x86, Atari, Amiga, Mac): Unix clone compatible with V7.

Sold with sources. Being POSIXified (sp?). For PCs, and surely

many others (eg. INMOS transputer). Newsgroup: comp.os.minix.

MipsOS: SVish (RISC/OS, now dropped, was BSDish)

more/BSD (VAX, HP 9000/300): Mt Xinu's Unix, based on 4.3BSD-Tahoe.

NCR UNIX: SVR4 (4.2?)

Net/2 tape (from Berkeley, 1991): BSD Unix, essentially compatible with

4.3BSD, includes only sources free of AT&T code, no low-level code.

See 386BSD and BSD/386 above.

NetBSD 0.8: is actually 386bsd in a new suit. Ported to [34]86, MIPS,

Amiga, Sun, Mac. What is relation to Net/2?

- 1.0 came out in '94.

NEXTSTEP (Intel Pentium and 86486, Hewlett-Packard PA-RISC, NeXT 68040):

BSD4.3 over Mach kernel, own GUI.

- 1.x, 2.0, 2.1, 2.2, 3.0, 3.1 (old)

- 3.2 (current version,

Intel Pentium and 86486,

Hewlett-Packard PA-RISC,

NeXT 68040)

- 3.3 (shipping; SPARC-version available)

- 4.0 (to be announced, will include Sun SPARC version and

will be OpenStep compliant

- no NEXTSTEP for PowerPC or DEC Alpha yet announced (are there plans?

NEWS-OS (Sony)

- 3.2

OSF/1 (DEC): DEC's port of OSF/1. I think this is now (4/93) available

on DEC's latest Alpha AXP (64-bit machine).

OSx (Pyramid): Dualport of both SysV.3 and BSD4.3. Newsgroup:

comp.sys.pyramid.

PC-IX (IBM 8086): SV

Plan 9 (AT&T): announced 1992, complete rewrite, not clear how close to

Unix it is. Key points: distributed, very small, various hardwares

(Sun, Mips, Next, SGI, generic hobbit, 680x0, PCs), C (not C++ as

rumors had it), new compiler, "8 1/2" window system (also very

small), 16-bit Unicode, CPU/file servers over high speed nets.

SCO Xenix (80x86): Versions for XT (not robust!), 286, 386 (with demand

paging). Today bulk of code is from System V. Stable product.

SCO Unix (80x86): SVR3.2 (stopped taking USL source at this point).

Sinix [Siemens]: System V base.

Solaris (Sparc, x86):

- 1.0: essentially same as SunOS 4.1.1, with OpenWindows 2.0 and

DeskSet utilities.

- 1.0.1: SunOS 4.1.2 with multiprocessing (kernel not multithreaded);

not for 386

- 2.0: (initially announced as SunOS 5.0 in 1988) based on SVR4

(with symmetric MP?), will include support for 386; with

OpenWindows 3.0 (X11R4) and OpenLook, DeskSet, ONC, NIS. Both

a.out (BSD) and elf (SVR4) formats. Kerberos support. Compilers

unbundled!

- Solaris is OpenStep compliant (non-NeXT, but with NEXTSTEP API)

with latest (1994?) version.

- Sun will ship its OpenStep-implementation with project DOE for

Solaris. First versions will be for SPARC-based Suns, but a

version for Solaris 2.4 for x86 and PowerPC will appear later.

SunOS (680x0, Sparc, i386): based on 4.3BSD, includes much from

System V. Main Sun achievements: NFS (1984), SunView (1985), NeWS

(1986, postscript imaging, now in OpenWindows), OpenLook GUI standard,

OpenWindows (NeWS, X11, SunView!). Newsgroup: comp.sys.sun.\*.

- 3.x: SV IPC package, FIFOs

- 4.0.3: lightweight processes, new virtual mem, shared libs

- 4.1: STREAMS & TLI, 8-bit clean?, async I/O, ms-dos file system

(continues as Solaris -- see above).

UHC (80x86): pure SVR4, X11, Motif

Ultrix (DEC): based on 4.2BSD with much of 4.3.

Newsgroup: comp.unix.ultrix.

- 4.4 is latest

UNICOS (Cray): System V base. Newsgroup: comp.unix.cray

- 5.x, 6,x, 7.0

UnixWare Release 4.2 [Univel]: SVR4.2; over NetWare. Univel no longer

exists.

UTEK (Tektronix)

- 4.0

VOLVIX (Archipel S.A.): UNIX-based OS built around a communication

based, distributed, real-time micro-kernel. SVR3.2 system calls,

BSD4.4 file/network system calls (VFS, FFS). Also NFS and X11.

Vanilla VOLVIX is for transputers.

Xenix (80x86): 1st Unix on Intel hardware, based on SVR2 (previously on

S III and even V7). Newsgroup: comp.unix.xenix.

---------------------------------------------------------------

Subject: Real-time Unices.

>From: "Pierre (P.) Lewis"

Date: Tue Aug 15 15:14:00 EDT 1995

X-Version: 2.9

6.7) Real-time Unices.

WARNING: this section is badly in need of work. It's full of errors,

and it's incomplete. I hope to have time to look at it this winter

(was "this fall"). I doubt all of following are Unices -- input is

welcome. The list also includes more common Unices with real-time

features, and some non-Unix systems with Unix-like APIs. I don't

suppose the latter really belong here, but having collected some notes,

I'm hesitant to junk them. See also comp.realtime.

AIX: AIX/6000 has real-time support.

Concurrent OS (Concurrent): real Unix, significantly modifed by

Concurrent.

CX/UX: a real UNIX significantly modified by Harris to provide

real-time capabilities and performance. Compliant with POSIX.4 final

version.

EP/LX (Control Data): port of LynxOS to R3000. Formerly called TC/IX.

LynxOS (Lynx Real-Time Systems, Inc): Berkeley and SV compatibility,

ground-up rewrite (proprietary), predates SVR4. Is not UNIX, but

supports much of the UNIX I/Fs (SV and BSD). POSIX compliant. Fully

preemptive, fixed priorities.

MiX: microkernel implementation of SVR4 offered by Chorus.

Motorola SVR4 has real-time capabilities.

QNX (Quantum Software): unix-compatible real real-time OS.

REAL/IX: based on System V 3.2 with RT features (fully premptive kernel,

fixed-priority scheduler, RT timer, &c.). For 68xxx and 88xxx based

systems. POSIX (1003.1 - 1988) compliant and in 88k form, it is

88open BCS compliant. Also available for x86/Pentium.

RTMX O/S [RTMX Incorporated]: elements of NET2, 4.4BSD-Lite and

proprietary code. Also includes FSF tools. Real-time (POSIX)

extensions.

RTU (Concurrent), for 68K boxes

Solaris 2 has real-time capabilities?

Stellix (Stardent); it's Unix, but is it real-time?

Venix/386: Interactive SVR3.2 with real-time extensions.

VMEexec (Motorola): not Unix, but also shares some I/Fs with Unix.

VxWorks (Wind River Systems): Little in common with Unix, has some I/Fs

in common with Unix (but not the file system). Newsgroup:

comp.os.vxworks.

(know nothing about)

Convex RTS

REAL/IX (AEG)

Sorix (Siemens)

System V/86 (Motorola)

TC/IX (CCD)

Velocity (Ready Systems):

---------------------------------------------------------------

Subject: Unix glossary.

>From: "Pierre (P.) Lewis"

Date: Tue Aug 15 15:14:00 EDT 1995

X-Version: 2.9

6.8) Unix glossary.

This section provides short definitions of various concepts and

components of (or related to) Unix systems.

Chorus: message-passing microkernel, may form basis for a future release

of SV. Chorus already have SVR4 running on top (binary-compatible).

CORBA (Common Object Request Broker Architecture).

COSE (Common Open Software Environment) [Sun, HP, IBM]: common look and

feel (Motif -- Sun will let OpenLook fade away), common API.

Reaction against Windows NT. See section 6.4 above.

DCE (Distributed Computing Environment, from OSF): Includes RPC (Apollo's

NCS), directory service (local based on DNS, global on X.500), time,

security, and threads services, DFS (distrib. file system), ....

OS-independent.

DME (Distributed Management Environment, from OSF): future.

DO (Distributed Objects [Enterprise]): ???.

FFS (Fast File System): from Berkeley, 1983. Equivalent (exact?) of

UFS in SunOS. Has notions such as cylinder groups, fragments.

FSF (Free Software Foundation)

Mach: modern kernels from CMU (Carnegie Mellon University) on which many

Unices and other OSs are based (e.g. OSF/1, MacMach, ...):

- 2.5: monolithic kernel with 4.2BSD

- 3.0: microkernel with BSD Unix server in user space (and other OSs,

e.g. MS-DOS)

Newsgroup: comp.os.mach

MFS (Memory File System):

NeWS (Network extensible Window System), from Sun?: PostScript-based,

networked, toolkits (and even clients) loaded in server. Part of

OpenWindows.

NFS (Network File System): contributed by Sun to BSD, stateless server

ONC (Open Network Computing): from Sun(?), includes RPC, name service

(NIS aka YP), NFS, ... (found in many Unices, other OSs).

OpenStep [NeXT, Sun]: ???

PowerOpen: both a standard, and an organization promoting PowerPC.

Involves IBM, Apple and Motorola; others? See section 6.4 above.

PowerPC (PPC): a RISC CPU chip [IBM, Motorola].

RFS (Remote File System): SV, stateful server, incompatible with NFS

RPC (Remote Procedure Call): high-level IPC (inter-process communication)

mechanism. Two flavors.

- ONC: Over TCP or UDP (later OSI), uses XDR to encode data.

- DCE: has a different RPC mechanism (based on Apollo's NCS)

S5 FS: System V's native file system, blocks 512 to 2K.

sockets: BSD interface mechanism to networks (compare TLI).

STREAMS: a message-passing kernel mechanism, initially in SVR3, which

provides a very good interface for protocol development.

TFS (Translucent File System): Sun, COW applied to files.

TLI (Transport Library Interface): SV's interface to transport services

(TCP, OSI). UI has also defined an APLI (ACSE/Presentation Library

Interface)

UFS (?): BSD's native file system as seen in SunOS, blocks 4K to 8K,

cylinder groups, fragments.

XTI (X/Open Transport Interface): TLI with enhancements

X11: pixel-oriented window system from MIT.

------------------------------

End of unix/faq Digest part 6 of 7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

Ted Timar - tmatimar@isgtec.com

ISG Technologies Inc., 6509 Airport Road, Mississauga, Ontario, Canada L4V 1S7

------------------------------------------------------------------------

Part1 - Part2 - Part3 - Part4 - Part5 - Part6 - Part7 - MultiPage

Rate this FAQ

[Vote]

Current Top-Rated FAQs

Not finding the answers you need? Submit your question for others to see.

Are you an expert in this area? Share your knowledge and earn expert

points by giving answers or rating people's questions and answers! This

section of FAQS.ORG is not sanctioned in any way by FAQ authors or

maintainers.

Questions strongly related to this FAQ:

\* i\'ma newbie ,can anybody please send me an article about FreeBSD

from instalation, how do... by [-NeWbiE-] (10/16/2003)

\* what is the difference bet. kill-15, kill-9 and other kill commands

in UNIX? by hari (7/14/2003)

\* I am a beginner in UNIX, I have accidentally put a large file in my

workspace and since... by nette (8/26/2003)

\* what is the featuers of UNIX OS by frank (8/15/2003)

\* How do I find the name of the current shell that I am working

on(Whether csh or ksh or... by mulla (10/27/2003)

\* how to create an environment variable by abc (9/14/2003)

\* How to download and upload files on UNIX Server from WIN2K server???

by Arvind (10/28/2003)

\* The Linux kernel does not allow paging out of kernel memory. What

effect does this... by shida (7/20/2003)

\* Using Microport System V release 4, can the Maximum Transmission Unit

be increased to... by Donn (7/21/2003)

Questions somewhat related to this FAQ:

\* I want to know under what condition(s) the following error will be

printed? I have a... by Vidya (7/29/2003)

\* HI ! I am compiling a program where perl is embedded in C on HPUX

64 bit... by Nipun Sharma (9/26/2003)

\* I am relatively new to UNIX and C++ programming, but I want to know

can you (1).... by vhc (6/27/2003)

Other questions awaiting answers:

\* 7054 questions related to other FAQs

\* 1211 general questions

\* 2238 answered questions

---------------------------------------------------------

[ Usenet FAQs | Search | Web FAQs | Documents | RFC Index ]

---------------------------------------------------------

Send corrections/additions to the FAQ Maintainer:

tmatimar@isgtec.com (Ted Timar)

Last Update December 12 2003 @ 00:51 AM

Part1 - Part2 - Part3 - Part4 - Part5 - Part6 - Part7 - MultiPage

------------------------------------------------------------------------

Unix - Frequently Asked Questions (7/7) [Frequent posting]

Not finding the answers

you need?

Submit your question for

others to see.

There are reader questions

on this topic!

Help others by sharing your

knowledge

------------------------------------------------------------------------

Message-ID:

X-Last-Updated: 1996/06/11

From: tmatimar@isgtec.com (Ted Timar)

Newsgroups: comp.unix.questions, comp.unix.shell

Subject: Unix - Frequently Asked Questions (7/7) [Frequent posting]

Date: 01 Dec 2003 09:14:56 GMT

Archive-name: unix-faq/faq/part7

Version: $Id: part7,v 2.9 1996/06/11 13:07:56 tmatimar Exp $

These seven articles contain the answers to some Frequently Asked

Questions often seen in comp.unix.questions and comp.unix.shell.

Please don't ask these questions again, they've been answered plenty

of times already - and please don't flame someone just because they may

not have read this particular posting. Thank you.

This collection of documents is Copyright (c) 1994, Ted Timar, except

Part 6, which is Copyright (c) 1994, Pierre Lewis and Ted Timar.

All rights reserved. Permission to distribute the collection is

hereby granted providing that distribution is electronic, no money

is involved, reasonable attempts are made to use the latest version

and all credits and this copyright notice are maintained.

Other requests for distribution will be considered. All reasonable

requests will be granted.

All information here has been contributed with good intentions, but

none of it is guaranteed either by the contributors or myself to be

accurate. The users of this information take all responsibility for

any damage that may occur.

Many FAQs, including this one, are available on the archive site

rtfm.mit.edu in the directory pub/usenet/news.answers.

The name under which a FAQ is archived appears in the "Archive-Name:"

line at the top of the article. This FAQ is archived as

"unix-faq/faq/part[1-7]".

These articles are divided approximately as follows:

1.\*) General questions.

2.\*) Relatively basic questions, likely to be asked by beginners.

3.\*) Intermediate questions.

4.\*) Advanced questions, likely to be asked by people who thought

they already knew all of the answers.

5.\*) Questions pertaining to the various shells, and the differences.

6.\*) An overview of Unix variants.

7.\*) An comparison of configuration management systems (RCS, SCCS).

This article includes answers to:

7.1) RCS vs SCCS: Introduction

7.2) RCS vs SCCS: How do the interfaces compare?

7.3) RCS vs SCCS: What's in a Revision File?

7.4) RCS vs SCCS: What are the keywords?

7.5) What's an RCS symbolic name?

7.6) RCS vs SCCS: How do they compare for performance?

7.7) RCS vs SCCS: Version Identification.

7.8) RCS vs SCCS: How do they handle problems?

7.9) RCS vs SCCS: How do they interact with make(1)?

7.10) RCS vs SCCS: Conversion

7.11) RCS vs SCCS: Support

7.12) RCS vs SCCS: Command Comparison

7.13) RCS vs SCCS: Acknowledgements

7.14) Can I get more information on configuration management systems?

If you're looking for the answer to, say, question 7.5, and want to skip

everything else, you can search ahead for the regular expression "^7.5)".

While these are all legitimate questions, they seem to crop up in

comp.unix.questions or comp.unix.shell on an annual basis, usually

followed by plenty of replies (only some of which are correct) and then

a period of griping about how the same questions keep coming up. You

may also like to read the monthly article "Answers to Frequently Asked

Questions" in the newsgroup "news.announce.newusers", which will tell

you what "UNIX" stands for.

With the variety of Unix systems in the world, it's hard to guarantee

that these answers will work everywhere. Read your local manual pages

before trying anything suggested here. If you have suggestions or

corrections for any of these answers, please send them to to

tmatimar@isgtec.com.

---------------------------------------------------------------

Subject: RCS vs SCCS: Introduction

Date: Sat, 10 Oct 92 19:34:39 +0200

>From: Bill Wohler <wohler@newt.com>

7.1) RCS vs SCCS: Introduction

The majority of the replies (in a recent poll) were in favor of

RCS, a few for SCCS, and a few suggested alternatives such as CVS.

Functionally RCS and SCCS are practically equal, with RCS having

a bit more features since it continues to be updated.

Note that RCS learned from the mistakes of SCCS...

---------------------------------------------------------------

Subject: RCS vs SCCS: How do the interfaces compare?

Date: Sat, 10 Oct 92 19:34:39 +0200

>From: Bill Wohler

7.2) RCS vs SCCS: How do the interfaces compare?

RCS has an easier interface for first time users. There are less

commands, it is more intuitive and consistent, and it provides

more useful arguments.

Branches have to be specifically created in SCCS. In RCS, they

are checked in as any other version.

---------------------------------------------------------------

Subject: RCS vs SCCS: What's in a Revision File?

Date: Sat, 10 Oct 92 19:34:39 +0200

>From: Bill Wohler

7.3) RCS vs SCCS: What's in a Revision File?

RCS keeps history in files with a ",v" suffix. SCCS keeps

history in files with a "s." prefix.

RCS looks for RCS files automatically in the current directory or

in a RCS subdirectory, or you can specify an alternate RCS file.

The sccs front end to SCCS always uses the SCCS directory. If

you don't use the sccs front end, you must specify the full SCCS

filename.

RCS stores its revisions by holding a copy of the latest version

and storing backward deltas. SCCS uses a "merged delta"

concept.

All RCS activity takes place within a single RCS file. SCCS

maintains several files. This can be messy and confusing.

Editing either RCS or SCCS files is a bad idea because mistakes

are so easy to make and so fatal to the history of the file.

Revision information is easy to edit in both types, whereas one

would not want to edit the actual text of a version in RCS. If

you edit an SCCS file, you will have to recalculate the checksum

using the admin program.

---------------------------------------------------------------

Subject: RCS vs SCCS: What are the keywords?

Date: Sat, 10 Oct 92 19:34:39 +0200

>From: Bill Wohler

7.4) RCS vs SCCS: What are the keywords?

RCS and SCCS use different keywords that are expanded in the

text. For SCCS the keyword "%I%" is replaced with the revision

number if the file is checked out for reading.

The RCS keywords are easier to remember, but keyword expansion is

more easily customized in SCCS.

In SCCS, keywords are expanded on a read-only get. If a version

with expanded keywords is copied into a file that will be

deltaed, the keywords will be lost and the version information in

the file will not be updated. On the other hand, RCS retains the

keywords when they are expanded so this is avoided.

---------------------------------------------------------------

Subject: What's an RCS symbolic name?

Date: Sat, 10 Oct 92 19:34:39 +0200

>From: Bill Wohler

7.5) What's an RCS symbolic name?

RCS allows you treat a set of files as a family of files while

SCCS is meant primarily for keeping the revision history of

files.

RCS accomplishes that with symbolic names: you can mark all the

source files associated with an application version with `rcs

-n', and then easily retrieve them later as a cohesive unit. In

SCCS you would have to do this by writing a script to write or

read all file names and versions to or from a file.

---------------------------------------------------------------

Subject: RCS vs SCCS: How do they compare for performance?

Date: Sat, 10 Oct 92 19:34:39 +0200

>From: Bill Wohler

7.6) RCS vs SCCS: How do they compare for performance?

Since RCS stores the latest version in full, it is much faster in

retrieving the latest version. After RCS version 5.6, it is also

faster than SCCS in retrieving older versions.

---------------------------------------------------------------

Subject: RCS vs SCCS: Version Identification.

Date: Sat, 10 Oct 92 19:34:39 +0200

>From: Bill Wohler

7.7) RCS vs SCCS: Version Identification.

SCCS is able to determine when a specific line of code was added

to a system.

---------------------------------------------------------------

Subject: RCS vs SCCS: How do they handle problems?

Date: Sat, 10 Oct 92 19:34:39 +0200

>From: Bill Wohler

7.8) RCS vs SCCS: How do they handle problems?

If you are missing the sccs or rcs tools, or the RCS or SCCS file

is corrupt and the tools don't work on it, you can still retrieve

the latest version in RCS. Not true with SCCS.

---------------------------------------------------------------

Subject: RCS vs SCCS: How do they interact with make(1)?

Date: Wed, 30 Dec 1992 10:41:51 -0700

>From: Blair P. Houghton <bhoughto@sedona.intel.com>

7.9) RCS vs SCCS: How do they interact with make(1)?

The fact that SCCS uses prefixes (s.file.c) means that make(1)

can't treat them in an ordinary manner, and special rules

(involving '~' characters) must be used in order for make(1) to

work with SCCS; even so, make(1) on some UNIX platforms will not

apply default rules to files that are being managed with SCCS.

The suffix notation (file.c,v) for RCS means that ordinary

suffix-rules can be used in all implementations of make(1), even

if the implementation isn't designed to handle RCS files

specially.

---------------------------------------------------------------

Subject: RCS vs SCCS: Conversion.

Date: Tue, 10 Jan 1995 21:01:41 -0500

>From: Ed Ravin <elr@wp.prodigy.com>

7.10) RCS vs SCCS: Conversion.

An unsupported C-Shell script is available to convert from SCCS

to RCS. You can find it in

ftp://ftp.std.com/src/gnu/cvs-1.3/contrib/

One would have to write their own script or program to convert

from RCS to SCCS.

---------------------------------------------------------------

Subject: RCS vs SCCS: Support

Date: Sat, 10 Oct 92 19:34:39 +0200

>From: Bill Wohler

7.11) RCS vs SCCS: Support

SCCS is supported by AT&T. RCS is supported by the Free Software

Foundation. Therefore RCS runs on many more platforms, including

PCs.

Most make programs recognize SCCS's "s." prefix while GNU make

is one of the few that handles RCS's ",v" suffix.

Some tar programs have a -F option that ignores either RCS

directories, or SCCS directories or both.

---------------------------------------------------------------

Subject: RCS vs SCCS: Command Comparison

Date: Sat, 10 Oct 92 19:34:39 +0200

>From: Bill Wohler

7.12) RCS vs SCCS: Command Comparison

SCCS RCS Explanation

==== === ===========

sccs admin -i -nfile file ci file Checks in the file

for the first time,

creating the revision

history file.

sccs get file co file Check out a file for

reading.

sccs edit file co -l file Check out a file for

modification.

sccs delta file ci file Check in a file

previously locked.

what file ident file Print keyword

information.

sccs prs file rlog file Print a history of

the file.

sccs sccsdiff -rx -ry file rcsdiff -rx -ry file Compare two

revisions.

sccs diffs file rcsdiff file Compare current with

last revision.

sccs edit -ix-y file rcsmerge -rx-y file Merge changes between

two versions into

file.

??? rcs -l file Lock the latest

revision.

??? rcs -u file Unlock the latest

revision. Possible

to break another's

lock, but mail is

sent to the other

user explaining why.

---------------------------------------------------------------

Subject: RCS vs SCCS: Acknowledgements

Date: Sat, 10 Oct 92 19:34:39 +0200

>From: Bill Wohler

7.13) RCS vs SCCS: Acknowledgements

I would like to thank the following persons for contributing to

these articles. I'd like to add your name to the list--please

send comments or more references to Bill Wohler .

Karl Vogel <vogel@c-17igp.wpafb.af.mil>

Mark Runyan <runyan@hpcuhc.cup.hp.com>

Paul Eggert <eggert@twinsun.com>

Greg Henderson <henders@infonode.ingr.com>

Dave Goldberg <dsg@mbunix.mitre.org>

Rob Kurver <rob@pact.nl>

Raymond Chen <rjc@math.princeton.edu>

Dwight <dwight@s1.gov>

---------------------------------------------------------------

Subject: Can I get more information on configuration management systems?

Date: Thu Oct 15 10:27:47 EDT 1992

>From: Ted Timar

7.14) Can I get more information on configuration management systems?

Bill Wohler, who compiled all of the information in this part of

the FAQ, has compiled much more information. This information is

available for ftp from ftp.wg.omron.co.jp (133.210.4.4) under

"pub/unix-faq/docs/rev-ctl-sys".

------------------------------

End of unix/faq Digest part 7 of 7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

Ted Timar - tmatimar@isgtec.com

ISG Technologies Inc., 6509 Airport Road, Mississauga, Ontario, Canada L4V 1S7

------------------------------------------------------------------------

Part1 - Part2 - Part3 - Part4 - Part5 - Part6 - Part7 - MultiPage

Rate this FAQ

[Vote]

Current Top-Rated FAQs

Not finding the answers you need? Submit your question for others to see.

Are you an expert in this area? Share your knowledge and earn expert

points by giving answers or rating people's questions and answers! This

section of FAQS.ORG is not sanctioned in any way by FAQ authors or

maintainers.

Questions strongly related to this FAQ:

\* i\'ma newbie ,can anybody please send me an article about FreeBSD

from instalation, how do... by [-NeWbiE-] (10/16/2003)

\* what is the difference bet. kill-15, kill-9 and other kill commands

in UNIX? by hari (7/14/2003)

\* I am a beginner in UNIX, I have accidentally put a large file in my

workspace and since... by nette (8/26/2003)

\* what is the featuers of UNIX OS by frank (8/15/2003)

\* How do I find the name of the current shell that I am working

on(Whether csh or ksh or... by mulla (10/27/2003)

\* how to create an environment variable by abc (9/14/2003)

\* How to download and upload files on UNIX Server from WIN2K server???

by Arvind (10/28/2003)

\* The Linux kernel does not allow paging out of kernel memory. What

effect does this... by shida (7/20/2003)

\* Using Microport System V release 4, can the Maximum Transmission Unit

be increased to... by Donn (7/21/2003)

Questions somewhat related to this FAQ:

\* I want to know under what condition(s) the following error will be

printed? I have a... by Vidya (7/29/2003)

\* HI ! I am compiling a program where perl is embedded in C on HPUX

64 bit... by Nipun Sharma (9/26/2003)

\* I am relatively new to UNIX and C++ programming, but I want to know

can you (1).... by vhc (6/27/2003)

Other questions awaiting answers:

\* 7054 questions related to other FAQs

\* 1211 general questions

\* 2238 answered questions

---------------------------------------------------------

[ Usenet FAQs | Search | Web FAQs | Documents | RFC Index ]

---------------------------------------------------------

Send corrections/additions to the FAQ Maintainer:

tmatimar@isgtec.com (Ted Timar)

Last Update December 12 2003 @ 00:51 AM

[ Usenet FAQs | Search | Web FAQs | Documents | RFC Index ]

Unix - Frequently Asked Questions (Contents) [Frequent posting]

Not finding the answers

you need?

Submit your question for

others to see.

There are reader questions

on this topic!

Help others by sharing your

knowledge

------------------------------------------------------------------------

Message-ID:

X-Last-Updated: 1996/06/11

From: tmatimar@isgtec.com (Ted Timar)

Newsgroups: comp.unix.questions, comp.unix.shell

Subject: Unix - Frequently Asked Questions (Contents) [Frequent posting]

Date: 01 Dec 2003 09:14:51 GMT

Archive-name: unix-faq/faq/contents

Version: $Id: contents,v 2.9 1996/06/11 13:08:13 tmatimar Exp $

The following seven articles contain the answers to some Frequently Asked

Questions often seen in comp.unix.questions and comp.unix.shell.

Please don't ask these questions again, they've been answered plenty

of times already - and please don't flame someone just because they may

not have read this particular posting. Thank you.

This collection of documents is Copyright (c) 1994, Ted Timar, except

Part 6, which is Copyright (c) 1994, Pierre Lewis and Ted Timar.

All rights reserved. Permission to distribute the collection is

hereby granted providing that distribution is electronic, no money

is involved, reasonable attempts are made to use the latest version

and all credits and this copyright notice are maintained.

Other requests for distribution will be considered. All reasonable

requests will be granted.

All information here has been contributed with good intentions, but

none of it is guaranteed either by the contributors or myself to be

accurate. The users of this information take all responsibility for

any damage that may occur.

Many FAQs, including this one, are available on the archive site

rtfm.mit.edu in the directory pub/usenet/news.answers.

The name under which a FAQ is archived appears in the "Archive-Name:"

line at the top of the article. This FAQ is archived as

"unix-faq/faq/part[1-7]".

These articles are divided approximately as follows:

1.\*) General questions.

2.\*) Relatively basic questions, likely to be asked by beginners.

3.\*) Intermediate questions.

4.\*) Advanced questions, likely to be asked by people who thought

they already knew all of the answers.

5.\*) Questions pertaining to the various shells, and the differences.

6.\*) An overview of Unix variants.

7.\*) An comparison of configuration management systems (RCS, SCCS).

The following questions are answered:

1.1) Who helped you put this list together?

1.2) When someone refers to 'rn(1)' or 'ctime(3)', what does

the number in parentheses mean?

1.3) What does {some strange unix command name} stand for?

1.4) How does the gateway between "comp.unix.questions" and the

"info-unix" mailing list work?

1.5) What are some useful Unix or C books?

1.6) What happened to the pronunciation list that used to be

part of this document?

2.1) How do I remove a file whose name begins with a "-" ?

2.2) How do I remove a file with funny characters in the filename ?

2.3) How do I get a recursive directory listing?

2.4) How do I get the current directory into my prompt?

2.5) How do I read characters from the terminal in a shell script?

2.6) How do I rename "\*.foo" to "\*.bar", or change file names

to lowercase?

2.7) Why do I get [some strange error message] when I

"rsh host command" ?

2.8) How do I {set an environment variable, change directory} inside a

program or shell script and have that change affect my

current shell?

2.9) How do I redirect stdout and stderr separately in csh?

2.10) How do I tell inside .cshrc if I'm a login shell?

2.11) How do I construct a shell glob-pattern that matches all files

except "." and ".." ?

2.12) How do I find the last argument in a Bourne shell script?

2.13) What's wrong with having '.' in your $PATH ?

2.14) How do I ring the terminal bell during a shell script?

2.15) Why can't I use "talk" to talk with my friend on machine X?

2.16) Why does calendar produce the wrong output?

3.1) How do I find the creation time of a file?

3.2) How do I use "rsh" without having the rsh hang around

until the remote command has completed?

3.3) How do I truncate a file?

3.4) Why doesn't find's "{}" symbol do what I want?

3.5) How do I set the permissions on a symbolic link?

3.6) How do I "undelete" a file?

3.7) How can a process detect if it's running in the background?

3.8) Why doesn't redirecting a loop work as intended? (Bourne shell)

3.9) How do I run 'passwd', 'ftp', 'telnet', 'tip' and other interactive

programs from a shell script or in the background?

3.10) How do I find the process ID of a program with a particular

name from inside a shell script or C program?

3.11) How do I check the exit status of a remote command

executed via "rsh" ?

3.12) Is it possible to pass shell variable settings into an awk program?

3.13) How do I get rid of zombie processes that persevere?

3.14) How do I get lines from a pipe as they are written instead of

only in larger blocks?

3.15) How do I get the date into a filename?

3.16) Why do some scripts start with #! ... ?

4.1) How do I read characters from a terminal without requiring the user

to hit RETURN?

4.2) How do I check to see if there are characters to be read without

actually reading?

4.3) How do I find the name of an open file?

4.4) How can an executing program determine its own pathname?

4.5) How do I use popen() to open a process for reading AND writing?

4.6) How do I sleep() in a C program for less than one second?

4.7) How can I get setuid shell scripts to work?

4.8) How can I find out which user or process has a file open or is using

a particular file system (so that I can unmount it?)

4.9) How do I keep track of people who are fingering me?

4.10) Is it possible to reconnect a process to a terminal after it has

been disconnected, e.g. after starting a program in the background

and logging out?

4.11) Is it possible to "spy" on a terminal, displaying the output

that's appearing on it on another terminal?

5.1) Can shells be classified into categories?

5.2) How do I "include" one shell script from within another

shell script?

5.3) Do all shells have aliases? Is there something else that

can be used?

5.4) How are shell variables assigned?

5.5) How can I tell if I am running an interactive shell?

5.6) What "dot" files do the various shells use?

5.7) I would like to know more about the differences between the

various shells. Is this information available some place?

6.1) Disclaimer and introduction.

6.2) A very brief look at Unix history.

6.3) Main Unix flavors.

6.4) Main Players and Unix Standards.

6.5) Identifying your Unix flavor.

6.6) Brief notes on some well-known (commercial/PD) Unices.

6.7) Real-time Unices.

6.8) Unix glossary.

6.9) Acknowledgements.

7.1) RCS vs SCCS: Introduction

7.2) RCS vs SCCS: How do the interfaces compare?

7.3) RCS vs SCCS: What's in a Revision File?

7.4) RCS vs SCCS: What are the keywords?

7.5) What's an RCS symbolic name?

7.6) RCS vs SCCS: How do they compare for performance?

7.7) RCS vs SCCS: Version Identification.

7.8) RCS vs SCCS: How do they handle with problems?

7.9) RCS vs SCCS: How do they interact with make(1)?

7.10) RCS vs SCCS: Conversion.

7.11) RCS vs SCCS: Support

7.12) RCS vs SCCS: Command Comparison

7.13) RCS vs SCCS: Acknowledgements

7.14) Can I get more information on configuration management systems?

If you're looking for the answer to, say, question 2.5, look in

part 2 and search for the regular expression "^2.5)".

While these are all legitimate questions, they seem to crop up in

comp.unix.questions or comp.unix.shell on an annual basis, usually

followed by plenty of replies (only some of which are correct) and then

a period of griping about how the same questions keep coming up. You

may also like to read the monthly article "Answers to Frequently Asked

Questions" in the newsgroup "news.announce.newusers", which will tell

you what "UNIX" stands for.

With the variety of Unix systems in the world, it's hard to guarantee

that these answers will work everywhere. Read your local manual pages

before trying anything suggested here. If you have suggestions or

corrections for any of these answers, please send them to to

tmatimar@isgtec.com.

--

Ted Timar - tmatimar@isgtec.com

ISG Technologies Inc., 6509 Airport Road, Mississauga, Ontario, Canada L4V 1S7

Rate this FAQ

[Vote]

Current Top-Rated FAQs

Not finding the answers you need? Submit your question for others to see.

Are you an expert in this area? Share your knowledge and earn expert

points by giving answers or rating people's questions and answers! This

section of FAQS.ORG is not sanctioned in any way by FAQ authors or

maintainers.

Questions strongly related to this FAQ:

\* what is the difference between korn shell and bourne shell by

priyanka (8/9/2003)

\* 1) How to rtfm? I've booted up cygwin, gentoo (in the system restore

CD), and PXES... by Briancady413 (10/21/2003)

\* How do we stop recursive find? i.e, if i put in a find command, it

should search for files... by VJ (7/25/2003)

\* How do I know which is my shell ? How do I set a default shell ? by

Sunny (7/22/2003)

\* what is the expansion for UNIX? by teetu (8/28/2003)

\* Dear Friends, In Solaris, I want to know the command to know the

current Shell that I am... by Yogesh (8/7/2003)

\* how to convert the # in the terminal window to $? by santosh

(8/4/2003)

Questions somewhat related to this FAQ:

\* When I tried to check out a symbolic link file using pvcs command

rget, it resulted in an... by chutki (7/11/2003)

\* If you were running Unix operating system on your computer, could you

use the same... by Michelle (8/29/2003)

\* What does PID and PPID columns mean when you use the "ps -a" after

using the "ps -f"... by Guicho (9/30/2003)

\* what is trans in Unix? by ramani (7/8/2003)

Other questions awaiting answers:

\* 7054 questions related to other FAQs

\* 1211 general questions

\* 2238 answered questions

---------------------------------------------------------

[ Usenet FAQs | Search | Web FAQs | Documents | RFC Index ]

---------------------------------------------------------

Send corrections/additions to the FAQ Maintainer:

tmatimar@isgtec.com (Ted Timar)

Last Update December 12 2003 @ 00:51 AM

IX. CHAPTER 7 TEXT EDITING WITH VI, EMACS AND SED.

Chapter 7 Text Editing with vi, EMACS and sed.

Page 169.

Vi is MODAL. It starts in Command Mode. i or a causes

it to enter Insert Mode.

Esc resumes Command Mode.

ESC:q to quit.

Create a file to edit with ls and redirection:

$ ls -l / > demo

$ vi demo

Practice hjkl movement commands.

Practice ^F, ^B, ^U and ^D page movement commands.

Insert mode: I, a, o, O.

o inserts a blank line below the line you're on and moves there.

O does the same thing, but on the line above the one you're on.

Commands for the vi editor.

Some advanced timesavers:

--------

vi editing multiple files: vi file1 file2 file3

write one file change to next:

:w

:n

Moving text from one file to another:

vi file1

(edit)

"f44Y (copy 44 lines to buffer f

:w

:e newfile

"fP insert the 44 lines into newfile.

--------

Simple vi command summary:

Object Change Delete Copy

1 word cw dw yw

2 words, not 2cw or c2w 2dw or d2w 2yw or y2w

counting punct

3 words back 3cb or c3b 3db or d3b 3yb or y3b

1 line cc dd yy or Y

to end of line c$ or C d$ or D y$

to beg of line c0 d0 y0

single char r x or X yl or yh

five chars 5s 5x 5yl

cursor to top cH dH yH

to bottom cL dL yL

to next line c+ d+ y+

to column 5 c5| d5| y5|

2d sentence 2c) 2d) 2y)

prev.paragraph c{ d{ y{

to pattern c/pattern d/pattern y/pattern

next pattern cn dn yn

end of file cG dG yG

line 13 c13G d13G y13G

Movement Commands

ldur h, j, k, l

forward by word w or W

backward by word b or B

to end of line $

to beg of line 0 or ^

Other ops

paste text from buffer p or P

no save, quit file :q!

Movement by screens:

^f scroll forward one screen

^b scroll backward one screen

^d scroll forward one-half screen

^u scroll backward one-half screen

Re-position screen but leave cursor on the current line:

z RETURN Move current line to top of screen

z . Move current line to center of screen

z - Move current line to bottom of screen

Movement within a screen

H cursor to top left (home) position

M cursor to middle line

L cursor to last line

RETURN first character of next line

+ first character of next line

- first character of previous line

9| move to column 9

( beginning of previous sentence

) beginning of next sentence

{ beginning of previous paragraph

} beginning of next paragraph

Searching

/pattern move to next occurrence of pattern

?pattern move to previous occurrence of pattern

n repeat search in same direction

N repeat search in opposite direction

/ repeat forward

? repeat backward

If you don't want the search to "wrap" back to the start

of the file, do :set nowrapscan

Read another file into the current position in this file

:r filename

Search-and-replace commands.

Print current line number and total lines in file: Control-G

. current line

0 end of file

$ end of file

+20 next 20 lines

% all lines in file

:/pattern/d delete next line containing pattern

:s/old/new/ replace first occurrance on this line of old with new

:s/old/new/g replace first occurrance on this line of old with new

:50,100s/old/new/g do likewise for lines 50-100

:1,$s/old/new/g do likewise for entire document

:%s/old/new/g same thing, since % means 1,$

:%s/old/new/gc global replacement with confirm

y RETURN means yes RETURN means no.

:map command

ESC :map keystroke command-line

Surround the current word with quotes:

ESC :map ^V^ai"^VESCea"ESC

"Map the following to control-a:

insert a pair of double quotes at present position,

press ESC press e to go to end of word, append a double-quote, re-enter command mode with ESC".

alternatively: use .exrc file to perminently customize vi.

:abbreviate command

ESC : abbreviate teh Thomas E. Harrisburg

Movement by line number.

To turn on line numbers do :set nu

To turn off line numbers do :set nonu

nG go to line n

`` (two backquotes) return to starting place before G

'' (two apostrophies) return to 0 on the starting line.

!! replace current line with a Unix command

!} replace current paragraph with the results of piping it through a Unix command

Buffers

The last 9 deleted lines are stored in buffers "1 - "9.

To recover the 2nd to last deletion type "2p

If you want to "step through" the deleted buffers undoing each paste

command, try:

"1pu.u.u and so on.

You can copy text into NAMED buffers a-z.

"dY or "dyy yank current line into buffer d

"a7yy yank next seven lines into buffer a

"x19dd cut next 19 lines into buffer x

"dP paste buffer d before cursor

"ap paste buffer a after cursor

If you specify the buffer name with a CAPITAL LETTER, the text you

cut or copy will be APPENDED to the text already in that buffer.

Bookmarks:

mc create a bookmark HERE called c

'c (apostrophie c) go to line containing bookmark c

`c go to character marked c

`` return to starting place

'' return to starting line

Saving part of a file:

:230,$w newname save line230 to EOF to file newname

:.,60w newname save from current line to line 60 as newname

:.,+60w newname save the next 60 lines as newname

Shell to UNIX: !

Chapter 7 - cont'd

--------

sed

'stream editor'

cat /etc/passwd | sed 's/ksh/The Korn Shell/ '

who | sed 's/tty/Logged in on device: /'

cat msg | sed 's/^/> /'

In vi, when you're replying to another's message saved in a file, you can:

!!cat file | sed 's/^/> /'

sed commands:

s substitute

d sed "1,5 d" infile ; delete lines 1 through 5 from output stream

i sed '/maison/i\

insert this line

' infile

; insert "insert this line" and a linefeed before line ; with "maison" in it

---------

Convert man command > command.man to regular text:

sed '

s/.^H//g

s/^[9//g

s/[sp\t]\*//g

(literal space or tab)

' filename

---------

The Vim editor is what we call "8-bit clean", that is, it can

edit BINARY files, even load modules. Of course, the user

has to know what she's doing, but consider the xxd utility.

xxd converts a file of binary to a hex dump format.

xxd -r (revert) makes it go back.

So, exit a.out, and when the screen comes up, type:

:%!xxd

(for all this file, execute xxd on the contents.

You'll see a dump. You can edit either the hex or the ascii.

Then, from command mode vi,

:%!xxd -r

Save file.

Be careful, but it does work.