

reference books

Running Linux and Practical C++ Both published by O'Reilly

A link to some of the books is here

<http://www.hk8.org/old%5Fweb/>

If you are part of some consortium (is BNL??- the UC is as well as many other universities)

<http://safari.oreilly.com/>

and look at Practical C++, as well as lots of other books.

ldd

to find out the libraries a library references

```
ldd /afs/rhic.bnl.gov/phenix/PHENIX_LIB/sys/i386_redhat80/new.4/lib/libfun4allfuncs.so
```

<lots of output>

c++filt

demangle C++ symbol

```
c++filt _ZNK7TObject7DoErrorEiPKcS1_Pc TObject::DoError(int, char const*, char const*, char*)  
const
```

nm – find out the dependencies in an object file

to install into your directory

```
autogen.sh --prefix=$MYINSTALLDIR
```

```
configure --prefix=$MYINSTALLDIR
```

To make nice eps from gif (or jpeg etc)

make a gif

then go to unix

```
imgtops -3 picture.gif > picture.eps
```

-3 means level 3 postscript

default is level 2

This compresses stuff nicely

get imgps from

<http://imgtops.sourceforge.net/>

OK now to build imgtops on cygwin its a pain

1) get python in cygwin

2) you need to build the python imaging library PIL

to do this

a) rebase the system

<http://www.tishler.net/jason/software/rebase/rebase-2.3.README>

get the tar file and bunzip and untar it

b) then download the PIL library etc as in the following

<http://www.cygwin.com/ml/cygwin/2003-06/msg01121.html>

I had to duplicate the the X11 directory to /usr/include/X11

Also see the following

<http://www.talkaboutprogramming.com/group/comp.lang.python/messages/312660.html>

3) then get imgtops and build it

Another way – use convert (I have had some trouble as convert seems not to make as small an eps file as one would like – e.g. the lanl archive –

Maybe I don't know the switches – rich)

Hi Rich,

I presume you are planning to add this to your "tips and tricks" book.

I didn't realize this is still a problem. I always use the swiss army knife of image formats, the Imagemagick toolbox. The "convert" utility will convert anything to anything, gif to eps for example, or anything to eps for that matter. IM is a bit more readily available. It can do a whole lot more - you can make animated gifs, automate the thumbnail generation from your foto album, automatically annotate images with, say, a date or some text, crop and transform. I have yet to see a transformation that you cannot accomplish, and, more importantly, script and so capture for re-use. It's at <http://www.imagemagick.org>. Just do

```
convert xx.gif xx.eps
```

That's it.

Now since with such a gif -> eps conversion you have the original gif (or jpg, or any other) file available, it's a good opportunity to make an EPS with a preview that you can import in, say, Powerpoint. This is where epstool comes in handy. You may recall my root macro "epssave.C" that we use to make eps files with a high-quality preview in much the same fashion. Where you start out with a gif and convert it to EPS, the root macro saves the primary graphic in both gif and eps, then we use epstool to combine the two into an EPS deluxe.

The macro tends to disappear from \$ROOTSYS/macros once in a while, I have a copy in my RCF home dir; I attach it here for your convenience.

[NOTE – for the tidbits readers – I put epssave.C in this area]

epstool is installed in RCF and is available at <http://www.cs.wisc.edu/~ghost/gsvieview/epstool.htm>

Fire up root and try this:

```
> root [0] .L epssave.C
> root [1] TH1F *x = new TH1F("test","test", 32,0,10) root [2]
> x->Draw();
> <TCanvas::MakeDefCanvas>: created default TCanvas with name c1 root
> [3] epssave(c1,"rich.eps"); Info in <TCanvas::Print>: GIF file
> /tmp/eps_29605.gif has been created Info in <TCanvas::Print>: eps file
> /tmp/eps_29605.eps has been created root [4] .q
```

The call takes the pointer to the canvas (c1) and a file name. The macro will show you how to do this outside of root. It also uses convert internally.

To remove a CVS lock

```
cd $CVSROOT/../../cvslocks/

..navigate to your directory...

rm the lockfiles you left dangling.
```

Tom

Anthony Denis Frawley wrote:

```
>Hi All,
>
```

> Can someone please remind me where I can find a CVS lock file so that
>I can remove it?

GDB hints

For starters on gdb go here:

http://www.hk8.org/old_web/linux/run/ch14_01.htm

If you debug code once in a while, you're probably (way too) familiar with the long time it takes for gdb to read in the symbols from the huge number of shared libraries we use. Here's how to disable the automatic reading of all those symbols and only read them in for the libraries you're interested in. Put this in your ~/.gdbinit file (or type it at the gdb prompt before you load any files):

```
set auto-solib-add off
```

and then type this for the libraries you are interested in

```
share REGEX
```

where REGEX is a regular expression that matches the libraries you want. Something like

```
share cgl
```

would load the symbols for libcgl.so. It would also load the libraries for libcglfoobar.so and anything else that matched the regular expression "cgl". This can be a big timesaver.

C++ questions

Hi folks,

Q: What's the difference between

```
void foo(const int & i);
```

and

```
void bar(const int i); ?
```

A: The "const" in the declaration of bar() is redundantly superfluous.

By default, C++ passes arguments by value, meaning that a copy is made of the argument and the subroutine then operates on that copy. In the second case, it's redundant to declare "i" const because bar() just can't possibly change the value of "i" as it appears in the calling routine. That's because bar() is operating on a copy of "i", so the original "i" is perfectly safe.

In the declaration of foo(), however, the const has some real effect. In foo(), the default "pass-by-value" mechanism is replaced with a Fortran-like "pass-by-reference". That means that foo() operates on the very same "i" that the calling routine uses. If foo() changes "i" then it is also changed in the calling routine.

We have a lot of this in our code. On picky compilers, for instance the OSF compiler, these constructs generate warnings like crazy. Pages and pages and pages of them.

Here's a command that does a pretty good job at finding these. A few false positives and a few false negatives, but not too bad. Cut and paste and hit return. Then edit your code.

```
find $OFFLINE_MAIN/include -name '*.h*' -exec \  
egrep -Hn '\([\^]*const *[\^&*]+[() ,]' {} \; | \  
grep -v gsl | grep -v CLHEP | grep -v boost | \  
grep -v _ref | grep -v oo
```

EMACS additions for c++

I think it might be worthwhile revisiting the topic of coding standards, but as a prelude to that particularly religious discussion, here are a couple of utilities for emacs that I've whipped up to help. Just copy the stuff below into your ~/.emacs file and restart emacs. Visit a header file, type "C-c 1" and the first function will wrap the body of that file in a "#ifdef ... #endif" block to prevent multiple inclusion. Visit any C/C++ source file, and each time you type "C-c 2", the second function will step a little further through your source code and offer simple suggestions for how you might make it more maintainable (e.g., add a pair of braces here, add a trailing ".0" to that number there - that sort of thing). Luckily for me, we don't have any coding standards defined for lisp!

Enjoy,
Dave

```
(defun c-cleanup ()  
  (interactive)  
  (let ((here))  
    (catch 'foo  
      (while  
        (re-search-forward  
          "\\(^[\\t ]*\\(if\\|while\\|else\\|for\\)\\)\\b\\|[a-zA-Z][0-9]\\.[^0-9]\\|  
          |[0-9]\\.[0-9]\\)"  
          (point-max) t)  
          (if (not (c-in-literal))  
              (progn  
                (setq here (point))  
                (cond ((progn  
                        (goto-char (match-beginning 0))  
                        (looking-at "[^\\t ]*for\\b"))  
                      (progn  
                        (c-end-of-statement)  
                        (c-end-of-statement))
```

```

        (re-search-forward ";" (point-max) t)
        (if (< (c-most-enclosing-brace (c-parse-state))
here)
            (progn
                (end-of-line)
                (message "Add {}'s around this statement")
                (throw 'foo t))))
        ((progn
            (goto-char (match-beginning 0))
            (looking-at "[\\t]*\\(if\\|while\\|else\\|\\b\"))
        (progn
            (re-search-forward ";" (point-max) t)
            (if (< (c-most-enclosing-brace (c-parse-state))
here)
                (progn
                    (end-of-line)
                    (message "Add {}'s around this statement")
                    (throw 'foo t))))
            ((progn
                (goto-char (match-beginning 0))
                (looking-at "[^a-zA-Z][0-9]\\.[^0-9]"))
            (progn
                (end-of-line)
                (message "Add a trailing `.0' to this number")
                (throw 'foo t)))
            ((progn
                (goto-char (match-beginning 0))
                (looking-at "[^0-9]\\.[0-9]"))
            (progn
                (end-of-line)
                (message "Add a prefix `0.' to this number")
                (throw 'foo t))))))
        (message "Everything looks OK!"))))

(defun c-include-buffer-once (&optional buffer)
  (interactive)
  (save-excursion
    (let ((s (buffer-name buffer)))
      (while (string-match "\\." s)
        (setq s (replace-match "_" t t s)))
      (setq s (concat "__" (upcase s) "__"))
      (goto-char (point-min))
      (insert "#ifndef ")
      (insert s)
      (newline)
      (insert "#define ")
      (insert s)
      (newline)
      (goto-char (point-max))
      (newline)
      (insert "#endif /* ")
      (insert s)
      (insert " */"))))

(add-hook 'c-mode-common-hook
  '(lambda ()
    (define-key c-mode-map "\C-c1" 'c-include-buffer-once)

```

```
(define-key c++-mode-map "\C-c2" 'c-cleanup))
```

More – make a c++ thing pretty – needs astyle

You can get astyle from astyle.sourceforge.net

If you add the lines below to your ~/.emacs file and restart emacs, you'll pick up a couple of new key bindings when you edit C or C++ code. Mark a region and type "C-c r" to use "astyle" to reformat that chunk of code, or just type "C-c b" to run astyle on the whole buffer. Astyle is a source code formatting program and it does a respectable job on C++ - much better than the current version of GNU indent which gets confused by all the angle bracket business of templates. The built-in emacs command "indent-region" only adjusts the leading whitespace of each line - astyle will do more than that. Astyle is available on all the RCF machines as /opt/phenix/bin/astyle.

```
(defvar astyle-command "astyle -p --style=gnu")

(defun astyle-region (start end)
  "Run astyle on region, formatting it in a pleasant way."
  (interactive "r")
  (save-excursion
    (shell-command-on-region start end astyle-command nil t)
    (indent-region start end nil)))

(defun astyle-buffer ()
  "Run astyle on whole buffer, formatting it in a pleasant way."
  (interactive)
  (save-excursion
    (astyle-region (point-min) (point-max))))

(add-hook 'c-mode-common-hook
  '(lambda ()
     (define-key c-mode-map "\C-cr" 'astyle-region)
     (define-key c-mode-map "\C-cb" 'astyle-buffer)
     (define-key c++-mode-map "\C-cr" 'astyle-region)
     (define-key c++-mode-map "\C-cb" 'astyle-buffer)))
```

Isnt this part of the cleanup?

For those code-jocks that use emacs, here's an updated version of a little widget I first wrote back when MJT was learning Fortran IV. Drop these lines into your ~/.emacs and you should be able to hit "C-c l" when editing a header file to insert a prophylactic set of #ifdef's at the top and bottom. Carpal-tunnel-be-gone. I've strengthened the regexps to handle the case of editing a duplicate buffer or one being edited over efs (aka ange-ftp to the old-school crowd).

```
;;; start here ...
```

```
(defun c-include-buffer-once ()
  (interactive)
  (save-excursion
    (let ((s
          (concat "__"
```

```

                (upcase
                 (substring
                  (buffer-name (current-buffer)) 0
                  (string-match "[<@$]" (buffer-name
(current-buffer))))))
                "__"))))

(while (string-match "\\\\" s)
  (setq s (replace-match "_" t t s)))

(goto-char (point-min))
(insert "#ifndef ")
(insert s)
(newline)
(insert "#define ")
(insert s)
(newline)
(goto-char (point-max))
(newline)
(insert "#endif /* ")
(insert s)
(insert " */"))))

(add-hook 'c-mode-common-hook
  '(lambda ()
    (define-key c-mode-map "\C-c1" 'c-include-buffer-once)
    (define-key c++-mode-map "\C-c1" 'c-include-buffer-once)))

;;; ... OK, that's enough

```

Hi all,

How about a little tutorial on sorting? We all do it, so we might as well do it well. Let's use an STL algorithm to sort some numbers. You may be thinking, "Oh no, I'm gonna have to use a vector or dqueue or some other crazy thing I've never heard of." Don't panic. Here's an example of how to sort an old-fashioned **array** of numbers:

```

#include <iostream>
#include <algorithm>

const int N = 5;
double a[N] = {2.1, 0.1, 4.1, 3.1, 8.1};

int
main()
{
  std::sort(a, a + N);

  for (int i = 0; i < N; i++)
  {
    std::cout << a[i] << std::endl;
  }
}

```

```
}
```

That's not that hard, right? Sometimes, you want to sort an array based on the values in another array - you know, produce an array of index values. A simple way to do that is to write a tiny class that holds two values: the index value and a way to get a value from the other array. Like this:

```
#include <iostream>
#include <algorithm>

const int N = 5;
double a[5] = {2.1, 0.1, 4.1, 3.1, 8.1};

struct tinyClass
{
    int i;
    bool operator<(const tinyClass& I) const
    {
        return a[i] < a[I.i];
    }
};

bool
print(const tinyClass &i)
{
    std::cout << "a[" << i.i << "] = " << a[i.i] << std::endl; }

int
main()
{
    tinyClass I[N];
    for (int i = 0; i < N; i++) {I[i].i = i;}
    std::sort(I, I + N);
    std::for_each(I, I + N, print);
}
```

This code makes an array of "tinyClass", fills it with stuff, and sorts it. Easy. This code also makes a couple of other points: one, don't be afraid of making tiny little classes that are only used in one place (not all classes are worthy of a separate header file and all that); and two, I assume that people know that "struct" is just like "class" except that everything in a struct is "public" by default.

Another way to do the same thing is to use the STL "pair" class, a small class which just holds two different things. If you sort a collection of pairs, the sort algorithm uses the first element of the pair to do the sorting, so we put the values we want to use for sorting there.

Here's an example with all the fancy stuff you might hope for:

```
#include <utility>
#include <vector>
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 5;
double a[5] = {2.1, 0.1, 4.1, 3.1, 8.1};

bool
print(const pair<double, int> &i)
{
    cout << "a[" << i.second << "] = " << i.first << endl; }

int
main()
{
    vector<pair<double, int> > v;
    for (int i = 0; i < N; i++)
        {
            pair<double, int> p(a[i], i);
            v.push_back(p);
        }
    sort(v.begin(), v.end());
    for_each(v.begin(), v.end(), print);
}
```

The next time you need to sort something, don't fret, it's simple.

Dave

Hi muonistas,

May I suggest that some enterprising muon'er recraft the little utility function MUTOO::SQUARE? It's one of those things that seems like a good idea at first glance - but isn't. Here's how it's implemented now:

```
double MUTOO::SQUARE(double x) { return x*x; }
```

Simple enough, but there are several problems with this. First it passes its argument by value, not by reference. This results in a (minor) performance hit of a few percent. Not that big a deal, I admit, but why throw away cycles needlessly? Here's a version that fixes that:

```
double MUTOO::SQUARE(double &x) { return x*x; }
```

The main problem, however, is that it is written for a single, fixed type argument: a double. If you call this function with a float (which is done all over the place), the compiler has to insert code to do the float to double conversion, and then it probably has to do the reverse to capture the result too. That's ridiculously slow.

The usual idiom for this sort of thing,

```
template <typename T>
static inline T
sqr(const T &x)
{
    return x * x;
}
```

it about *ten* times faster (compared to calling the original version with a float argument). In the kind of code we write, squaring and square-rooting are very common operations, so this should be fixed.

Notes on jobs sizes:

Hi folks,

How can I say this delicately? Some analysis programs are just too damn big to run on the computers in the RCF. In fact, they're too damn big to run on most any computer you're likely to find. I've cc'ed this note directly to a few habitués of the mile-high RAM club. The machines in the RCF have 1 GB of memory. They are set up to run two batch jobs. If you have a job that uses a lot more than 500 MB (more is OK, a lot more is not so OK) then the machine will just degenerate into a useless room heater doing a slow thiorazine shuffle as it swaps pages in and out and in and out.

The jobs I'm talking about are using 3 GB of memory. Not 600 MB or 900 MB or even 1 GB. 3 GB.

It's probably as simple as some otherwise useful array that has accreted a few new dimensions. Or maybe there's a nasty memory leak. I don't know.

The problem is that the machine becomes useless not just for the person who submitted a big, big, big job, but also for the other person whose batch job had the misfortune to find itself on the same machine.

Dave

```
> Hi Dave
> Is there an easy way for us to see this in our batch jobs?
> -r
```

Hi Rich,

Didja look at the options for "bjobs"?

"The -l option displays the following additional information:
project name, job command, current working directory on the sub-mission
host, pending and suspending reasons, job status, resource usage,
resource usage limits information."

So, to take one of Hubert's current jobs as an example:

```
rcas2078% bjobs -r -u hubert -q phenix_cas | tail -1
379945 hubert RUN phenix_cas rcas2067 rcrs2125 AuAu114836 Mar
18 18:11
rcas2078% bjobs -l 379945
```

[... buncha blah, blah, blah elided ...]

```
Thu Mar 24 11:37:06: Resource usage collected.
The CPU time used is 11399 seconds.
IDLE_FACTOR(cputime/runtime): 0.04
MEM: 427 Mbytes; SWAP: 2035 Mbytes; NTHREAD: 5
PGID: 1864; PIDs: 1864 1867 1868 1869 1870
```

There ya go. 2GB of swap. [RKS note - 2GB is too big]

Here I check a jobs of mine

```
rcas2077 /direct/phenix+data08/seto/run4/cabana/Out % bjobs -l | more

Job <529197>, Job Name <Outewg47435>, User <seto>, Project <default>, Status
<RUN>, Queue <phenix_cas>, Job Priority <50>, Command
</phenix/data08/seto/run4/cabana/jobfile.csh filelist.txt.435
out_electron435.root>
Wed Mar 23 11:44:12: Submitted from host <rcas2077>, CWD
</direct/phenix+data08/seto/run4/cabana/Out/Outewg47>, Output File
<log435.log>, Requested Resources <linux>;
Thu Mar 24 05:40:33: Started on <rcrs2125>, Execution Home </phenix/u/seto>,
Execution CWD </direct/phenix+data08/seto/run4/cabana/Out/Ou
tewg47>;
Thu Mar 24 12:17:08: Resource usage collected.
The CPU time used is 279 seconds.
IDLE_FACTOR(cputime/runtime): 0.24
MEM: 155 Mbytes; SWAP: 830 Mbytes; NTHREAD: 5
PGID: 947; PIDs: 947 948 949 1012 1013
```

SCHEDULING PARAMETERS:

	r15s	r1m	r15m	ut	pg	io	ls	it	tmp	swp	mem
loadSched	-	0.7	-	-	-	-	-	-	-	-	-
loadStop	-	2.0	-	-	-	-	-	-	-	-	-

Here the SWAP<1GB. Its Ok.

To: phenix-off-l@bnl.gov

Subject: RE: subfigure

I guess most people don't usually program in TeX, but here's a little tidbit that should help a little bit. Put this near the top of your next LaTeX document:

```
\def\mean#1{\left<#1\right>}
```

and then, when you want to indicate a mean value in a formula, do it like this:

```
$$\mean{x^2}$$
```

(supposing you want to indicate the mean value of x squared) Now you'll get properly sized angle brackets that tightly surround both the "x" and the superscript "2" instead of a too small "less than" sign and a too small "greater than" sign too far away. Looks better. Easier to read.

It's the right thing to do.

Richard Seto
Physics Dept
University of California
Riverside CA, 92620
951-827-5623 Fax(4529) [new]
richard.seto@ucr.edu

> -----Original Message-----

> From: owner-phenix-off-1@bnl.gov

> [mailto:owner-phenix-off-1@bnl.gov] On Behalf Of David Morrison

> Sent: Tuesday, February 08, 2005 11:23 AM

> To: PHENIX-OFF-L@bnl.gov

> Subject: subfigure

>

> Hi all,

>

> OK, another LaTeX tip. I've seen people use lots of ways to

> group multiple figures into one for a paper. Lots of

> "minipage" and so on.

> Here's a simpler way - just use the "subfigure" package

> that's part of a standard LaTeX distribution. Suppose you've

> got a collection of figures nobel1.eps, ..., nobel4.eps that

> you want to show as a group. Do something like this:

>

```
> \documentclass{article}
```

```
> \usepackage{graphicx}
```

```
> \usepackage{subfigure}
```

```
> \begin{document}
```

```

>
> \begin{ figure }[htbp]
> \centering
> \subfigure[Top left figure]{
> \label{fig:tl}
> \includegraphics[width=0.45\textwidth]{nobel1.eps} }
> \hspace{0.05\textwidth}
> \subfigure[Top right figure]{
> \label{fig:tr}
> \includegraphics[width=0.45\textwidth]{nobel2.eps} }
> \vspace{0.05\textwidth}
> \subfigure[Bottom left figure]{
> \label{fig:bl}
> \includegraphics[width=0.45\textwidth]{nobel3.eps} }
> \hspace{0.05\textwidth}
> \subfigure[Bottom right figure]{
> \label{fig:br}
> \includegraphics[width=0.45\textwidth]{nobel4.eps} }
> \caption{Caption for the whole collection.}
> \label{fig:all}
> \end{ figure }
>
> \end{ document }
>
> And they all end up nice and neat with individual captions
> and labels.
> Spiffy.
>
> Dave
>

```

Cout doesn't work problem

If you find your cout etc commands don't work, you will need to include the line using namespace std; in your code

Comment on Job size which should be < 1MB
Hi Rich,

Probably Dave has some other, better and less obvious tricks up his sleeves, and I'm sure his invitation would apply to you too or anyone else for that matter--actually I was "happy" with my analysis code's size of ~600 MB and admittedly was being squeaky about other people's larger code ;>. In the past I've had the most success in reducing the size of my code not by modifying the dimensions of arrays, but reducing the component objects that the arrays are arrays of. This usually means making custom classes which store the minimal amount of info.

Another tidbit (in fact just in this spirit) that I just learned about the other day for people who use CabanaBoy is the use of the "UltraLight" classes. (See [offline/AnalysisTrain/UltraLight](#)). This base class provides a fast, simple, and standardized way to convert any class into one that inherits from PHParticle and is as bare bones as possible. This allows you in CB, rather than storing many buffers of PHCentralTrackvX's (which are HUGE because of the hundreds of data fields on this class), instead to store (much smaller) buffers of your own "UltraLight" paired down version of PHCentralTrack. It also has the nice benefit that it provides a standard way to convert ANY class into a PHParticle and therefore use CabanaBoy. For example, this is probably the best way to use CabanaBoy with Emcal Clusters currently, since the node tree emc object emcClusterContainer does not inherit from PHParticle.