Computer Programming I Quick Reference / Bookmark (BYUH, IS230, Spring 2001)

http://is230.byuh.edu/

Professor

Name: Don Colton
Email: don@colton.byuh.edu
Office: GCB 130B

Tutoring

Hours: Mon-Fri, 8–10 PM
Where: GCB 140

Secure Telnet Client

Connect to **is230.byuh.edu**
Under Microsoft Windows use TeraTerm **ttssh**
(available at is230.byuh.edu)

Unix Commands

fg right before logging out
jobs see what jobs are still running
exit or **logout** to log out
passwd change your password
gcc Gnu C Compiler
gcc abc.c compile abc.c
cat (concatenate) type out a file
ls -l list (long format) the directory
mv move (rename) a file
rm remove (delete) a file
mkdir make a new directory
cd change directory
pwd print working (current) directory
rmdir remove a directory
du disk usage (lists directory tree)
df disk freespace
w see who is logged in
top see why machine is slow
Delete-key delete prior character
Backspace-key not always = delete
info xyz learn how to use xyz
man xyz learn how to use xyz

Emacs Commands

emacs (from unix) start the editor
M- (meta) press Esc, then release
C- (ctrl) means hold down Ctrl
C-g abandon the command in progress
C-x C-c exit emacs (see also C-z)
C-h t tutorial: learn emacs better

C-x C-f edit a file (maybe new)
C-x C-s save your changes
C-x C-w write: save-as (rename)
C-x i insert text from another file
C-x b bounce between buffers
C-x C-b see a list of buffers
C-x 1 (one) reduce screen to one window

C-p go to previous line (cursor up 1)
C-n go to next line (cursor down 1)
C-f go forward one position
C-b go backward one position
C-d delete current character
Delete-key delete prior character
C-a go to start of line
C-e go to end of line
C-k kill (cut) to end of line
kill several times to cut several lines
C-y yank (paste) back what was killed
C-k C-y kill/yank (copy)
C-s search (type one letter at a time)
C-_ undo last change
C-x u undo last change
M-C-f go to matching close paren/brace
M-C-b go to matching open paren/brace
C-l (el) center and redraw screen

Shell Out / Compile / Test

C-x C-w dem1-1.c save your file
C-z put something to sleep temporarily
this is also called "shelling out"
fg (foreground) wakes it up again
C-x C-c exit emacs permanently
gcc dem1-1.c compile
a.out run it (test it)
C-c kill what's running
fg foreground: wake up emacs

Submitting / Rmail

M-x rmail (from emacs) do email
q quit rmail, go back to emacs
C-x C-c quit rmail and emacs
m start a new email message
To: GradeBot@gradebot.byuh.edu
Subject: is230 labname submit a lab
Subject: is230 status for status report
BEGIN on line before program (optional)
make sure nothing else is on that line
C-x i insert the program you want to send
END on line after program (optional)
make sure nothing else is on that line
do **not** use any attachments
do **not** use any fancy encoding
C-c C-c send the message
after sending wait five seconds...
g get new mail
C-s ### search for trouble
C-l (el) stop the search right here
n go to next message
p go to previous message
d delete current message
u undelete prior deleted message
c continue editing an email
C-x 1 (one) reduce screen to one window
C-x 2 split screen into two windows
C-x o jump to next (other) window

C Required Changes

Always declare "main" to return an int.
`int main ( ) {`
Make your output end on a fresh line.
`printf("\n");` (if needed)
Make your exit status informative.
`return 0;` (from main)
`exit(0);` (from anywhere)

Highly Recommended

Use `\n` at the end (not start) of print lines.
Put `{` on same line as main/if/do/while
Indent several spaces for each `{`
Unindent the same for each `}`

printf

`%c` for a character
`%d` for a decimal number
`%s` for a string (char arrays)
`%f` for a float or double
number inside specifies field width
`-` left justifies
`0` fills with leading zeros
neither (default) right justifies
`+` puts `+` on positive numbers
␣ leaves room for a sign on positives
neither (default) puts no sign on positives
`+` and `-` are **NOT** related
anything before `%` prints as is
anything after d/s/f prints as is

Precedence

```
* / %
+ -
< <= > >=
== !=
&&
||
```

Integer Division

The division example is `x` cookies and `y` children.
Give as many cookies as you can to each child.
Always give the same number to each child.
Never break cookies (makes children sad).
`x / y` how many for each child?
`x % y` how many left for mom and dad?

Truth Tables (and / or)

```
T && T = T
? && F = F
F && ? = F

T || ? = T
? || T = T
F || F = F

zero -> false
non-zero -> true
false -> zero
true -> one
```

Increment / Decrement

make sure to get the order right
`++x` add, save, use new value
`--x` subtract, save, use new value
`x++` use old value, but add, save new value
`x--` use old value, but subtract, save
the order controls what's used inside a calculation
the order **never matters** outside of the calculation

Common C Errors

warning: return-type defaults to 'int'
you forgot `int` before `main`

warning: control reaches end of non-void function
you forgot `return 0;`

warning: implicit declaration of function 'printf'
you forgot `#include <stdio.h>`

parse error before `'<'`
maybe you forgot `#` before `include`

warning: implicit declaration of function 'pintf'
undefined reference to 'pintf'
you misspelled `printf`

25: parse error before 'yada'
missing semi-colon before yada on line 25

1: studio.h: No such file or directory
you misspelled `stdio.h`

Notes