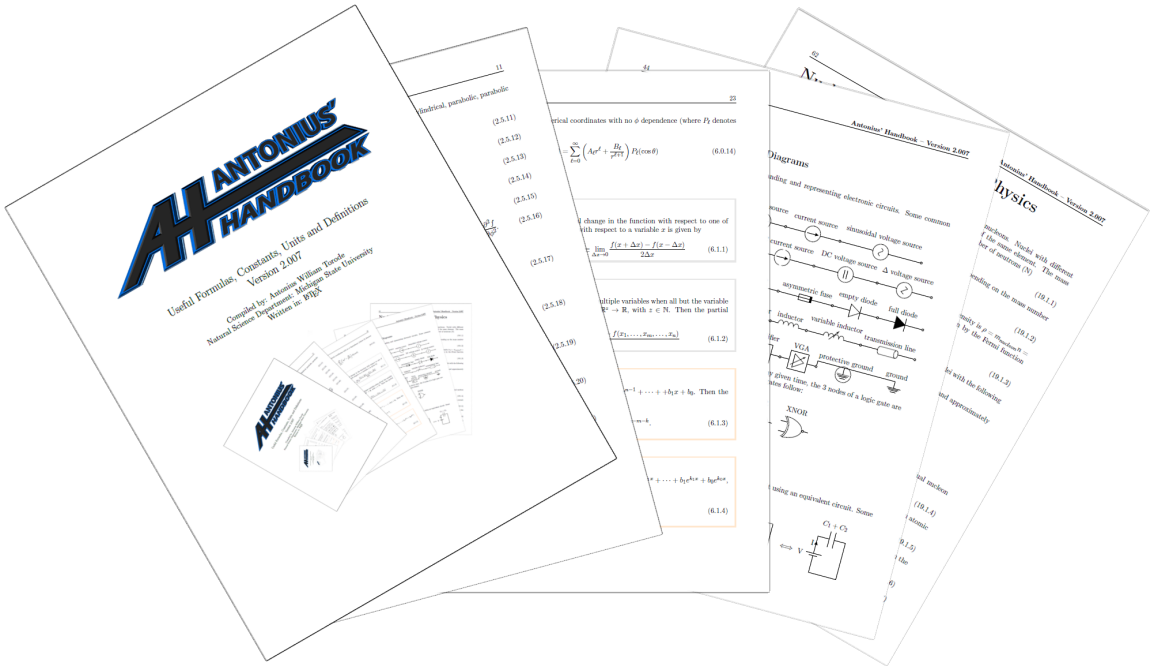
ANTONIUS' HANDBOOK

Useful Formulas, Constants, Units and Definitions
Volume II - Programmers Paradise
Version 0.005

Compiled by: Antonius William Torode
Natural Science Department: Michigan State University
Written in: LaTeX



The original maintainer of this work is: Antonius Torode.

The current maintainer of this work is: Antonius Torode.

Published by Antonius Torode.

Hosted at: https://msu.edu/~torodean/AHandbook.html

Github Repository: https://github.com/torodean/Antonius-Handbook

First Personal Release (Version 0.000):	June 2017
First Public Release (Version 1.000):	N/A
Most Current Revision Date (Version 0.005):	April 10, 2018

Torode, A.
Antonius' Handbook.
Michigan State University –
Department of Physics & Astronomy.
2016, Student.
Volume II.
Version: 0.005

Preface

This document is a compilation of useful programming formulations, definitions, constants, and general information used throughout my own schooling and research as a reference while furthering education. It's purpose is to provide a complete 'encyclopedia' per say of various codes, syntax and significant ideas used often. The idea and motivation behind it is to be a quick reference providing easily accessible access to necessary information for either double checking or recalling proper formulations or algorithms for use in various situations due to my own shortcomings in matters of memorization. All the material in this document was either directly copied from one of the references listed at the end or derived from scratch. On occasion typos may exist due to human error but will be corrected when discovered.

The version number is updated every time the document is distributed, printed, or referred to. This ensures that there is no two copies with different information and similar version numbers. The latest update date is automatically set to the current date each time the document is edited. Please refrain from distributing this handbook without permission from the original author/compiler. This book is formatted for printing.

For more information about this book or details about how to obtain your own copy please visit:

https://msu.edu/~torodean/AHandbook.html

Disclaimer

This book contains codes, formulas, definitions, and theorems that by nature are very precise. Due to this, some of the material in this book was taken directly from other sources. This is only such in cases where a change in wording or codes could cause ambiguities or loss of information quality. Following this, all sources used are listed in the references section.

This page intentionally left blank.
(Yes, this is a contradiction.)

Contents

1 Introduction **1**

2 Linux **2**
2.1 System Related Commands . . . 2
2.2 Files and Storage . . . 2
2.3 Users and Groups . . . 3
2.4 Networking . . . 3
2.5 Shell Scripting . . . 4

3 Mac **6**

4 C++ **7**
4.1 Basic Input and Output . . . 7
4.1.1 Simulate Key Strokes (Windows Only) . . . 7
4.2 Variable Types . . . 8
4.2.1 Converting Between Types . . . 8
4.3 Mathematical Commands . . . 8
4.4 System Commands . . . 9
4.5 Qt Specific . . . 9

5 ROOT **10**

6 Python **11**
6.1 Plotting and Graphs . . . 11

7 Resources **12**

References **12**

Introduction

This document is still under the initial formatting stages and useful information will be added soon. When it has sufficient information to be ready for distribution the version will be updated to 1.000.

Linux

Linux is a broad subcategory that encompass a large family of free and open sourced operating systems. Installing, setting up, and using a linux based operating system is the perfect way for anyone to gain knowledge, understanding, and practice of how a computer system truly works. Unlike the end user experience with Windows and Mac OS, linux has a much higher capability for customization and a higher degree of freedom. With that said, linux is not necessarily more user friendly to the new or average computer user, however it is free in most cases!

2.1 System Related Commands

Retreive information and valid arguments for a command.

```
COMMAND --help    # COMMAND must be a valid command such as cd, ls, etc...
```

Changing directory via terminal

```
cd /directory # Changes the directory to the subdirectory /directory
cd ..         # Goes back one directory
```

How to display the processes that are currently running.

```
ps aux
```

To search the results of a command for a string of characters one can use the grep command. For example:

```
ps aux | grep "firefox"
```

Restore power/battery icon if it disappears.

```
/usr/lib/x86_64-linux-gnu/indicator-power/indicator-power-service &disown
```

Restore volume icon/control button if it disappears.

```
gsettings set com.canonical.indicator.sound visible true
```

Reset wifi services in case the connection gets lost.

```
sudo systemctl restart network-manager.service
```

Turn off LCD display.

```
xset dpms force off //turns off display.
```

Change or view the host name of a computer with the hostname file.

```
sudo nano /etc/hostname # Opens this file using nano for editing.
hostname                # Command to see what the current hostname is.
```

Make a file executable and execute a file

```
chmod a+x /location/of/FILE # Makes a file executable
./FILE                      # Executes a file.
```

2.2 Files and Storage

Copy a file or directory to a different computer

```
# To copy a file.
scp -v <File Path> username@computer:"<path to copy to>"

# To copy a directory.
scp -rv <File Path> username@computer:"<path to copy to>"
```

Show information about the file system on which each FILE resides, or all file systems by default.

```
df
```

List information about File(s) (in the current directory by default).

```
ls        # list all items in a directory
ls -1     # list all items in a directory (one item per line)
ls -lh    # list all items in a directory with size, owner, and date modified
```

List all of the block devices (hence partitions) detected by the machine

```
lsblk
```

Mount and unmount a partition

```
sudo mount <DEVICE TO MOUNT> <MOUNT POINT>
sudo mount /dev/sdb1/ /mnt/       # example of mounting
sudo umount <DEVICE TO MOUNT> <MOUNT POINT>
sudo umount /dev/sdb1/ /mnt/      # example of mounting
```

2.3 Users and Groups

Create a new user

```
sudo useradd [options] <USERNAME>              #Creates a user
sudo useradd -e 2016-02-05 <NAME>              #Creates a user that expites on a day.
sudo useradd <USERNAME> -G <GROUPNAME>         #Adds a user to a group upon creation.
useradd --help                                 #See full useradd options.
```

Change a users password

```
passwd <USERNAME>
```

Change the user in terminal

```
su - <USERNAME>
```

Add a user to the sudoers group

```
usermod -aG sudo <USERNAME>
```

2.4 Networking

The **ifconfig** command is for viewing IP configuration information and configuring network interface parameters.

```
ifconfig
```

The **traceroute** command is for printing the route that packets take to a network host.

```
traceroute
```

The **Domain Information Groper** is used to perform DNS lookups and display answers returned from the DNS servers.

```
dig
```

The **telnet** command connects the destination host:port via the telnet protocal. An established connection means connectivity between two hosts is properly working.

```
telnet
```

The **nslookup** command is for querying Internet domain name servers.

```
nslookup
```

The **netstat** command is used to review open network connections and open sockets.

```
netstat
```

The **nmap** command is used to check for opened ports on a server

```
nmap <SERVER NAME>
```

The **ifup** and **ifdown** commands are used to disable network interfaces.

```
# enables an ethernet parameter
ifup <ETHERNET INTERFACE PARAMETER>
ifup eth0  # example: enables 'eth0'

# disables an ethernet parameter
ifdown <ETHERNET INTERFACE PARAMETER>
ifdown eth0  # example: disables 'eth0'
```

Enable/Disable **IPv6**. This is only a temporary solution as it may turn itself back on after some time.

```
#Use these two commands to disable IPv6
sudo sysctl -w net.ipv6.conf.all.disable_ipv6=1
sudo sysctl -w net.ipv6.conf.default.disable_ipv6=1

#Use these two commands to re-enable IPv6
sudo sysctl -w net.ipv6.conf.all.disable_ipv6=0
sudo sysctl -w net.ipv6.conf.default.disable_ipv6=0
```

2.5 Shell Scripting

To create a shell script you must create a new text file and save it as a '.sh' file. The file should start with the directory to the proper shell which is generally the default below. The first line (starting with '#!') is not a comment, but instead is treated by Unix as "which shell do I use to run this code." In our case, the Bourne shell will be used [3].

```
#!/bin/sh
# This is a comment!
```

To print text one can use the **echo** command as follows.

```
#!/bin/sh
echo Hello World
echo "Hello World"
```

To make a file executable, the **chmod** command can be used and is typically used as follows.

```
chmod a+rx <SCRIPTNAME>.sh
```

Shell script **variables** are created by use of the equal sign. spaces in lines containing variables need to be avoided. To reference a variable, the '$' character is used. Quotations are used to avoid ambiguities with spaces.

```
#!/bin/sh
MY_VARIABLE="Hello World"     # Creates a variable.
echo $MY_VARIABLE             # Prints the variable.
```

The **touch** command can be used to create a new empty file.

```
#!/bin/sh
echo "What is your name?"
read USER_NAME
echo "Hello $USER_NAME"
echo "I will create you a file called ${USER_NAME}_file"

# The quotations prevent multiple files from being called to touch.
touch "${USER_NAME}_file"
```

Mac

3.1: Mac Startup Options

Mac has various startup features. To use them, hold the following keys down simultaneously upon startup as soon as you hear the startup chime:

Startup Keys	Descriptions
Command, R	Boot into OS X Recovery mode.
C	Boot to external device such as CD, DVD, or USB.
N	Netboot.
Shift	Safe Boot.
Command, V	Boot using verbose mode for comprehensive boot details.
Command, S	Single user mode.
Command, Option, P, R	Resetting the PRAM during boot.
T	Enable target disk mode.

C++

4.1 Basic Input and Output

To output text via a terminal you can use:

```
std::string text = "Hello World!";
std::cout << text << std::endl; //std::endl is equivalent to the new-line character.
```

To get input as a user in the type of a std::string, you can use:

```
std::string input = "";
std::cout << "Enter some text: ";
std::getline(std::cin, input);
```

4.1.1 Simulate Key Strokes (Windows Only)

First the correct files must be included and an event must be setup.

```
#define WINVER 0x0500
#include <windows.h>

INPUT ip;

ip.type = INPUT_KEYBOARD; // Set up a generic keyboard event.
ip.ki.wScan = 0; // hardware scan code for key
ip.ki.time = 0;
ip.ki.dwExtraInfo = 0;
```

After this, functions can be setup to simulate various keys based on the specific key codes, two examples of such are

```
void space(){
   // Press the "space" key.
   ip.ki.wVk = VK_SPACE; // virtual-key code for the "space" key.
   ip.ki.dwFlags = 0; // 0 for key press
   SendInput(1, &ip, sizeof(INPUT));

   // Release the "space" key
   ip.ki.wVk = VK_SPACE; // virtual-key code for the "space" key.
   ip.ki.dwFlags = KEYEVENTF_KEYUP; // KEYEVENTF_KEYUP for key release
   SendInput(1, &ip, sizeof(INPUT));
   Sleep(50);
}

void one(){
   // Press the "1" key.
   ip.ki.wVk = 0x31; // virtual-key code for the "1" key.
   ip.ki.dwFlags = 0; // 0 for key press
   SendInput(1, &ip, sizeof(INPUT));

   // Release the "1" key.
   ip.ki.wVk = 0x31; // virtual-key code for the "1" key.
   ip.ki.dwFlags = KEYEVENTF_KEYUP; // KEYEVENTF_KEYUP for key release.
   SendInput(1, \&ip, sizeof(INPUT));
   Sleep(50);
}
```

A similar method can be used to simulate mouse clicks. And example for left click follows

```
void leftclick(){
   INPUT ip={0};
   // left down
   ip.type = INPUT_MOUSE;
   ip.mi.dwFlags = MOUSEEVENTF_LEFTDOWN;
   SendInput(1,&Input,sizeof(INPUT));

   // left up
   ZeroMemory(&Input,sizeof(INPUT));
   ip.type = INPUT_MOUSE;
   ip.mi.dwFlags = MOUSEEVENTF_LEFTUP;
   SendInput(1,&Input,sizeof(INPUT));
   Sleep(50);
}
```

4.2 Variable Types

Creating and using a vector.

```
#include <vector>

int size1 = 5;
int size2 = 6;

//Creates a vector named V1 containing int's with a size of 5 and sets each element to 0.
std::vector<int> V1(size1, 0);

//Creates a 2-D vector (vector containing vectors) of size 5x6 named V2 containing doubles;
std::vector< std::vector<double>> V2(size1, std::vector<double>(size2, 0));

V1[0] = 8; //Sets the first element in V1 to 8.

V2[0][3] = 3.1415; //Sets the 4th element in the first row of V2 to 3.1415.
```

4.2.1 Converting Between Types

std::string to int

To convert a string to an integer you can use the **stoi** function:

```
std::string text = "31415";
int number = std::stoi(text);
```

std::string to double

To convert a string to a double you can use the **stod** function:

```
std::string text = "3.1415";
double number = std::stof(text);
```

4.3 Mathematical Commands

Prime Number

A simple brute for method to determines if a number of type long is prime or not.

```
bool isPrime(long num) {
   int c = 0;   //c is a counter for how many numbers can divide evenly into num
   if (num == 0 || num == 1 || num == 4) {
      return false;
   }
   for (long i = 1; i <= ((num + 1) / 2); i++) {
      if (c < 2) {
         if (num % i == 0) {
            c++;
         }
      } else {
         return false;
      }
   }
   return true;
}
```

4.4 System Commands

Sleep

Make the thread sleep for some amount of time using the std::chrono to determine the duration [1].

```
#include <thread>
#include <chrono>

std::this_thread::sleep_for(std::chrono::milliseconds(50)); //Makes the system sleep for 50
    milliseconds.

std::this_thread::sleep_for(std::chrono::seconds(50)); //Makes the system sleep for 50
    seconds.
```

On a Windows specific program this can be simplified by including the windows.h header

```
#include <windows.h>

Sleep(50); //Makes the system sleep for 50 milliseconds.

Sleep(5000); //Makes the system sleep for 50 seconds.
```

ROOT

Formatting a TString can be simplified by use of the Form function.

```
const char *someText = "Hello World!";
int someInt = 2314;

//%s corresponts to a const char*
//%d corresponds to an integer
TString output = Form("I want to say %s a total of %d times", someText, someInt);

//This will output "I want to say Hello World! a total of 2314 times"
std::cout << output;
```

Python

The official python documentation can be found at the following links

```
# Documentation for version 3+
https://docs.python.org/3/
# Documentation for version 2+
https://docs.python.org/2/
```

Import floating point division which allows python 2 compatibility when using division with doubles. Include this at the beginning of the script.

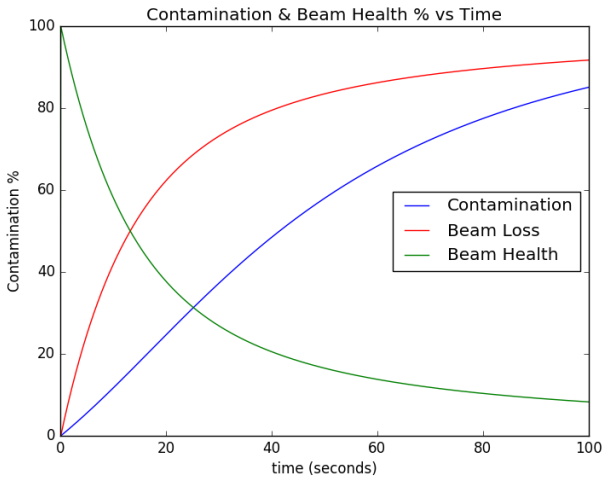
```
from __future__ import division
```

6.1 Plotting and Graphs

A nicely formatted plot with a legend using the pylab package.

```
import pylab as plt #Imports the correct packages for plotting.

plt.title('Contamination & Beam Health % vs Time') # Creates a title.

plt.plot(t, Contamination, '-b', label='Contamination') #Plots Contamination in blue.
plt.plot(t, Beam_loss, '-r', label='Beam Loss') #Plots Beam_loss in red.
plt.plot(t, Beam_health, '-g', label='Beam Health') #Plots Beam_health in green.
#plt.plot(x,y,'-color', label='Legend Label') #Template

plt.xlabel("time (seconds)") #Creates a x-axis label
plt.ylabel("Contamination %") #Creates a y-axis label

plt.legend(loc='center right') #Creates a legend with the labels set above.
#Other locations include upper/lower/center left/right

plt.show() #Displays plot.
```

This code would display a graph such as the one below such that the proper values are input.



References

[1] http://www.cplusplus.com/reference/chrono/

[2] Kumar, Chandan. 10 Useful Linux Networking Commands. Geek Flare, 11 Feb. 2018, geekflare.com/linux-networking-commands/.

[3] Parker, Steve. Shell Scripting Tutorial. The Shell Scripting Tutorial, www.shellscript.sh/.

Index

Converting Between Types, 8

dig, 3

echo, 4

ifconfig, 3
Input and Output, 7
IPv6, 4

netstat, 4
Networking, 3
nmap, 4
nslookup, 4

Plotting and Graphs, 11
Prime Number, 8

Simulate Key Strokes, 7
Sleep, 9
std::string to int, 8
stod, 8
stoi, 8
System Commands, 9
System Related Commands, 2

telnet, 4
touch, 5
traceroute, 3

Variable Types, 8
variables, 4
Vector, 8