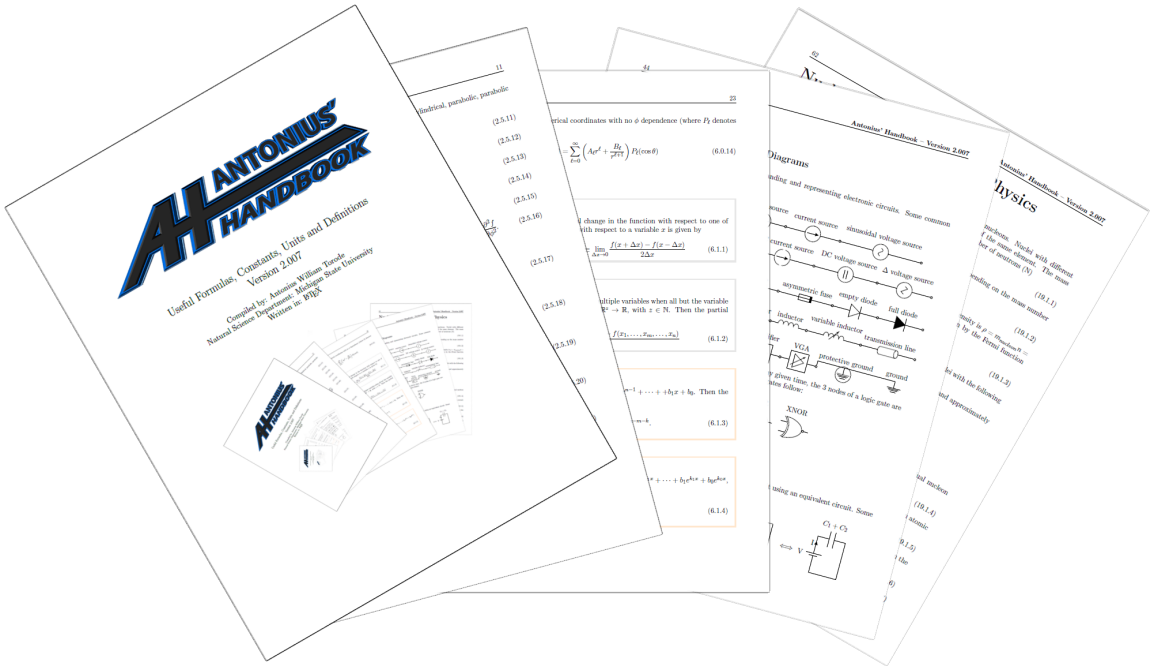
ANTONIUS' HANDBOOK

Useful Formulas, Constants, Units and Definitions
Volume II - Programmers Paradise
Version 0.002

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



© 2016 Antonius Torode
All rights reserved.

This work may be distributed and/or modified under the conditions of Antonius General Purpose License (AGPL).

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 0.002):	August 17, 2017

Torode, A.
Antonius' Handbook.
Michigan State University –
Department of Physics & Astronomy.
2016, Student.
Volume II.
Version: 0.002

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

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.

C++

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);
```

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);
}
```

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);

//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));

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.
```

Converting Between Types

std::string to int

To convert a string to an integer you can use:

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

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;
}
```

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.
```

Linux

System Related Commands

Retreive information and valid arguments for a command.

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

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
```

Changing directory via terminal

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

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

```
df
```

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
```

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.
```

Networking

View IP configuration information

```
ifconfig
```

Enable/Disable IPv6

```
#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
```

Python

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

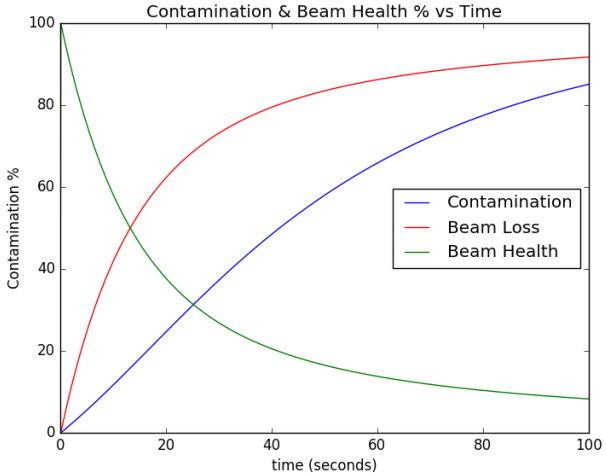
```
from __future__ import division
```

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/

Index