| R · I · T | Rochester Institute of Technology<br>Golisano College of Computing and Information Sciences<br>Department of Information Sciences and Technologies |
|---|---|

**4002-XXX**
**Software Development on Linux Systems**
**Lab 07 – Documentation**

**Name:** _____ **Section:** _____

**Overview**

In this lab you will be building documentation for developers, end users and support. This will include code documentation, end user documentation and licensing documentation. You will be building man pages and text documents that you will include in the final package.

**The first man page activity of this lab is the only section that must be done individually and it requires a sign off. The rest may be done as a group, but each group member should work on each part to familiarize themselves with it.**

**The licensing section MUST be done as a project group, unless you are not in a group.**

**Overview – Man Page**

Man pages are quick manuals built as references for commands, libraries and software packages. These provide local documentation on how to use package, the arguments it takes, example usage, where bugs can be reported and further information. These should be concise, but clear and complete.

Man pages are broken into several sections based on their category. Though many categories exist, the standard categories are:

| Category | Description |
|---|---|
| 1 | General Commands |
| 2 | System Calls |
| 3 | Library Functions and subroutines |
| 4 | Devices and drivers |
| 5 | File formats and convention |
| 6 | Games |
| 7 | Miscellaneous |
| 8 | System Administration and daemons |

Inside of each man page you will find a number of information sections. These are some of the typical sections you will find in a man page:

| Category | Description |
| --- | --- |
| Name | This is the name of the program and a one list description of it |
| Synopsis | This is the base syntax on how to invoke the program from the command line |
| Description | This is a clear thought out description of the program, its functionality and purpose |
| Options | This is a list of parameters and flags that the program can take along with what they do |
| Bugs | This is a URL and/or email that the user can reference to report bugs and find bugs. This may also be a list of known bugs. |
| See Also | This is a list of other documentation (such as html manuals/tutorials) and related programs the user may be interested in. |

Some other optional categories include author, version, known bugs, history, examples, files, exit status and environment.

R · I · T

Rochester Institute of Technology
Golisano College of Computing and Information Sciences
Department of Information Sciences and Technologies

Example of a man page:

NAME
hello - Application that prints hello world

SYNOPSIS
hello [--debug] [-n #]

DESCRIPTION
hello 1.1

hello allows you to print “Hello World” from the command. This is done as practice in many languages as the first application written. Hello-world is written in C++ and allows invocation from the command line.

By default this program does not have a graphic user interface, but could easily be embedded into one.

OPTIONS

--debug
Print verbose runtime information to the terminal that can be used for debugging purposes.

-n
When followed by a number this is the number of times hello world will be printed to the terminal. By default this number is always 1. This flag allows it to be repeated for a specified number of times. This must be followed by a number or else it will fail. The maximum number that can be specified is 100.

BUGS
See http://myDebuggingSiteForHello.org.net.edu for a current list of bugs.

Bugs may also be reported at the link above or emailed to myDebug@debug.org.net.edu.edu.mail.net.org

SEE ALSO
gcc(1), g++(1), goodbye(1)

R · I · T

Rochester Institute of Technology
Golisano College of Computing and Information Sciences
Department of Information Sciences and Technologies

**Overview – Man Page Syntax**

Man pages follow a special formatting for each section. The sections are the Title Header (.TH), Section Header (.SH) and Subsection (.SS). There are also formatting options, such as bold font (.B), italic font (.I) and item paragraph (.IP). These are case sensitive and will not work if incorrect. The escape character for symbols in man pages is \ . For example **\-** will produce a hyphen in the document.

Breakdown of command syntax:
**.TH [program name] [section number] [center footer] [left footer] [center header]**

Example: **.**TH foo 1 "January 2011" "1.1 Linux Release" "User manuals"

This will create a header at the top of the document that reads
**foo(1) User Manuals foo(1)**

and a footer that reads
**1.1 Linux Release January 2011 foo(1)**

**.SH [section name]**

Example: .SH NAME
foo **\-** Converts data into foo.

This will produce a new section called name that reads
**NAME**
foo – Converts data into foo.

**.SS [subsection name]**

This works the same as .SH, but it will be indented and bold under the current section. All data under this section will be indented one level further.

Example: .SH OPTIONS
.SS "FILE OPTIONS"
These are my file options.

This will produce a section and subsection that reads
**OPTIONS**
**FILE OPTIONS**
These are my file options

**.IP [item name]**

The itemized paragraph can be used to create an itemized list or used individually for style.

Example: .SH OPTIONS
.IP --debug
Print verbose runtime information to the terminal that can be used for debugging purposes.

This will produce a section that reads
**OPTIONS**
**--debug**
Print verbose runtime information to the terminal that can be used for debugging purposes.

**.B [word or words]**

Example: This is text that will produce
.B bold text
on the screen.

This will produce text that reads
This is text that will produce **bold text** on the screen.

**.I [word or words]**

.I works the same as .B, but will produce italic text.

| R · I · T | Rochester Institute of Technology<br>Golisano College of Computing and Information Sciences<br>Department of Information Sciences and Technologies |
|---|---|

**Activity – Building Man Pages (MUST BE DONE INDIVIDUALLY)**

Using the information in the overviews, you will create a simple man page with the base sections – **name, synopsis, description, options, bugs and see also**. You may use any Linux text editor for this task. You will need to reproduce the following example. Header and footer information have been enclosed to differentiate them from the rest of the example. You do not need to worry about syntax for hyperlinks. The beginning of the code is given on the next page.

| hellotest(1) | User manuals | hellotest(1) |
|---|---|---|

**NAME**

hellotest - Application that prints "hello world "

**SYNOPSIS**

hellotest [--debug] [-n #]

**DESCRIPTION**

**hellotest 1.1**

hellotest allows you to print **"Hello World"** from the command. This is done as practice in many languages as the first application written.

**OPTIONS**

**--debug**

Print verbose runtime information to the terminal that can be used for debugging purposes.

**-n**

When followed by a number this is the number of times hello world will be printed to the terminal.

**BUGS**

See http://myDebuggingSiteForHello.org.net.edu for a current list of bugs or email new bugs to myDebug@debug.org.net.edu.edu.mail.net.org

**SEE ALSO**

**gcc(1), g++(1), goodbye(1)**

| 1.1 Linux Version | January 2011 | hellotest(1) |
|---|---|---|

| R · I · T | Rochester Institute of Technology<br>Golisano College of Computing and Information Sciences<br>Department of Information Sciences and Technologies |
|---|---|

Code

```
.TH hellotest 1 "January 2011" "1.1 Linux Version" "User Manuals"
.SH NAME
hellotest \- Application that prints "hello world"
.SH SYNOPSIS
hellotest [--debug][-n #]
.SH DESCRIPTION
.B hellotest 1.1

hellotest allows you to print “Hello World” from the command. This is done as
practice in many languages as the first application written.
.SH OPTIONS
.IP --debug
Print verbose runtime information to the terminal that can be used for
debugging purposes.
```

1) Paste the code into a text editor
2) Save the file as “hellotest**.**1”
3) Open terminal and cd to the directory you saved the file
4) You may test your file with “ man **./**hellotest.1 ”

**SAVE THESE FILES AS YOU WILL BE USING THEM IN THE NEXT LAB**

**Instructor/TA Sign-Off** _____

**Activity – Building Man Pages II**

Now you will build a man page for your project. You want it to be as long as necessary, but as short as possible. This means make the information complete, but be concise and do not include unnecessary information, such as how the software works internally, unless it is necessary for the user to utilize the software. You will turn this in for credit for this section of the lab.

You will need to have these sections -

- **name**
- **synopsis**
- **description**
- **bugs.**
- If your project takes any command line arguments, you must provide an **options** section.
- If your project has more than one command that invokes it, your project must provide a **see also** section.

| R · I · T | Rochester Institute of Technology<br>Golisano College of Computing and Information Sciences<br>Department of Information Sciences and Technologies |
|---|---|

**Overview – Change Logs**

In this activity you will be building change logs. A change log is a list of recent changes to your software. Change logs are formatted differently between Fedora software and Ubuntu software. Fedora packages follow the Fedora change log format, while Ubuntu packages follow the Debian change log format.

Debian Format
**NOTE: THIS FILE IS EXTREMELY SENSITIVE; SPACING MUST BE RIGHT**

*Package (version) distribution; urgency=urgency*
***[two spaces]**** *change details*
***[four spaces]*** *more change log details*

***[two spaces]***other change details*
***[one space]**--maintainer-name**[one space]**<email>* ***[two spaces]*** *date*

Explanation

| Tag | Purpose | Options |
|---|---|---|
| Package | Software name | Name |
| version | Version number | Number(s) |
| distribution | Package type (determined by team) | Experimental, stable, stable-security, testing-proposed-updates, testing-security, unreleased, unstable, ***name***<br><br>***name*** may be a release name the team have chosen, such as penguin, coyote, fossil, etc |
| urgency | Severity type (determined by team) | Low, medium, high, emergency, critical<br><br>*emergency and critical are treated the same |
| maintainer-name | Real name or pseudonym | Real name or pseudonym |
| date | Date in RFC 2822 and RFC 5322 format | Day-of-week, dd month yyyy hh:mm:ss +**zzzz**<br><br>***zzzz** is the time zone in UTC time which is HHMM. In New York this is **-0500** |

Debian Example

hellotest (1.1) unstable; urgency=low

*New version released.
fixed compile errors on Linux Kernel 2.8
*Added new music
New songs are owned by the team and free to use
Playback can be done in mono or stereo
*Added goodbye-world to install dependencies
*Printing changes for efficiency

-- MaintainerGuy <maintainer@mGuy.it.com.edu> Fri, 12 Jan 2012 14:23:20 -0500

hellotest (1.0) stable; urgency=medium

*Modified code to prevent sporadic crashes
*Removed midi sound component
*Added gcc to install dependencies
+Requires gcc (<= 4.5.2)
+Requires lib-sdl (<= 1.2.14)

-- MaintainerGuy <maintainer@mGuy.it.com.edu> Mon, 20 Dec 2011 15:50:43 -0500

Fedora Format

** day-of-week mmmm dd yyyy Name <email> - [version]*
- *comment*
- *comment*

Fedora Example

* Thu Jan 19 2012 MaintainerGuy <maintainer@mGuy.it.com.edu> 1.1
- New version released
- Fixed compile errors on Linux Kernel 2.8

* Wed Jan 18 2012 MaintainerGuy <maintainer@mGuy.it.com.edu> 1.0
- Modified code to prevent sporadic crashes
- Removed midi sound component

| R · I · T | Rochester Institute of Technology<br>Golisano College of Computing and Information Sciences<br>Department of Information Sciences and Technologies |
|---|---|

**Activity – Building Change Logs**

Now you will need to build change logs for an application called hellotest. In this exercise you will create a file called changelog that has the following contents. You also need to make the Fedora format equivalent. **Ensure the spacing is correct.**

hellotest (1.4-2) unstable; urgency=low

  * updated code to with less ram
  * now saved files are encrypted

 -- programmerGuy <programmerGuy@mkdir.net.com.url>  Tue, 24 Jan 2012 18:48:11 -0500

hellotest (1.4-1) unstable; urgency=low

  * Initial release

 -- programmerGuy <programmerGuy@mkdir.net.com.url>  Sun, 22 Jan 2012 18:48:11 -0500

**Now make the Fedora equivalent using the contents above with the file name changelog.fedora**

**SAVE THESE FILES AS YOU WILL BE USING THEM IN THE NEXT LAB**

| R · I · T | Rochester Institute of Technology<br>Golisano College of Computing and Information Sciences<br>Department of Information Sciences and Technologies |
|---|---|

**Overview – License Files**

Everyone open source project that has a license must include it with the source code. This is very strict when including other open source code or libraries. Licensing is handled differently between Fedora and Ubuntu packages. Fedora follows the Fedora package licensing format, while Ubuntu follows the Debian package licensing format.

The main licenses are available on MyCourses in the section for this lab. If your project is not licensed to one of these, you likely can find your license here.
http://www.opensource.org/licenses/category

Debian Format:

Format: http://dep.debian.net/deps/dep5
Upstream-Name: <package name>
Source: <url://example.com>

Files: *
Copyright: [one space]<years> <put author's name and email here>
[one space]<years> <likewise for another author>
License: <license name>
[one space]Put the license text here **indented by 1 space**

Files: debian/*
Copyright:[one space]<years> <put author's name and email here>
License: <license name>
[one space]Put the license text here **indented by 1 space**

**NOTICE THERE IS ONE SPACE INDENTATION FOR EACH COPYRIGHT AUTHOR AND FOR EACH LINE OF LICENSE TEXT**



Example:

```
Format: http://dep.debian.net/deps/dep5
Upstream-Name: hello
Source: url://example.com

Files: *
Copyright:  2012  programmerGuy  <programmerGuy@guy.net.edu>
            2012 programmerFriend  <programmerFriend@guy.net.edu>

License: GPL-2+
 This package is free software; you can redistribute it and/or modify
 it under the terms of the GNU General Public License as published by
 the Free Software Foundation; either version 2 of the License, or
 (at your option) any later version.
 This package is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.
 .
 You should have received a copy of the GNU General Public License
 along with this program. If not, see <http://www.gnu.org/licenses/>
 .
 On Debian systems, the complete text of the GNU General
 Public License version 2 can be found in "/usr/share/common-licenses/GPL-2".

Files: debian/*
Copyright:  2012  programmerGuy  <programmerGuy@guy.net.edu>

License: GPL-2+
 This package is free software; you can redistribute it and/or modify
 it under the terms of the GNU General Public License as published by
 the Free Software Foundation; either version 2 of the License, or
 (at your option) any later version.
 .
 This package is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.
 .
 You should have received a copy of the GNU General Public License
 along with this program. If not, see <http://www.gnu.org/licenses/>
 .
 On Debian systems, the complete text of the GNU General
 Public License version 2 can be found in "/usr/share/common-licenses/GPL-2".
```

Fedora Format:

For Fedora, the licensing information is specified in the packaging configuration (SPEC) file. You list the license you are using and list the location of the license document you are providing. Then you would add the license file to the package with the file name COPYING.

These are the lines in the SPEC file:
License: license-name
%doc COPYING

Example 1:

License: GPL
%doc COPYING

**Activity – License Files**

Download the appropriate license for your project. In most cases these will probably be in the lab folder. Otherwise, you may have to find and provide the document yourself.

1. Build a Debian formatted COPYRIGHT file using the information from the license you downloaded. Be extremely cautious of spacing.

2. Rename the downloaded license file to COPYING (for fedora format).

| R · I · T | Rochester Institute of Technology<br>Golisano College of Computing and Information Sciences<br>Department of Information Sciences and Technologies |
|---|---|

**Overview – README Files**

README files are files that contain end user instructions, configuration information and general information users need to know. After the header, there is not necessarily a format for a README file to follow. You may provide general information, such as a description, controls, contact information or other information you consider important. Similarly, you may provide end user instructions, configuration information, customization options or other setup/configuration options.

The top of your README file should contain the program name, version, document copyright info, date and author. After the header, you may decide which information is necessary to the end user.

Example 1 README header:

> hello-world version 1.1
> Copyright January 2011 programmerGuy <progGuy@infoMix.it.it.edu.ca.zol>

**Activity – README Files**

Build a README file for your project as part of the write up for this lab. You will have to determine the contents of the README yourself. You will also turn in a more complete version with your final project.

**Activity – Code Commenting**

As open source code is seen by many eyes, it is important that it is well commented. If you have actively been commenting your code, then this will not take much time. Make sure all of your class files are commented so that people who are new to the project can understand what is happening and be able to improve it or modify it for their needs.

Also, make sure your license header is at the top of each file (.c, .cpp, .java, etc). It is important to have the license headers in each file for reference and proof that the file is licensed.

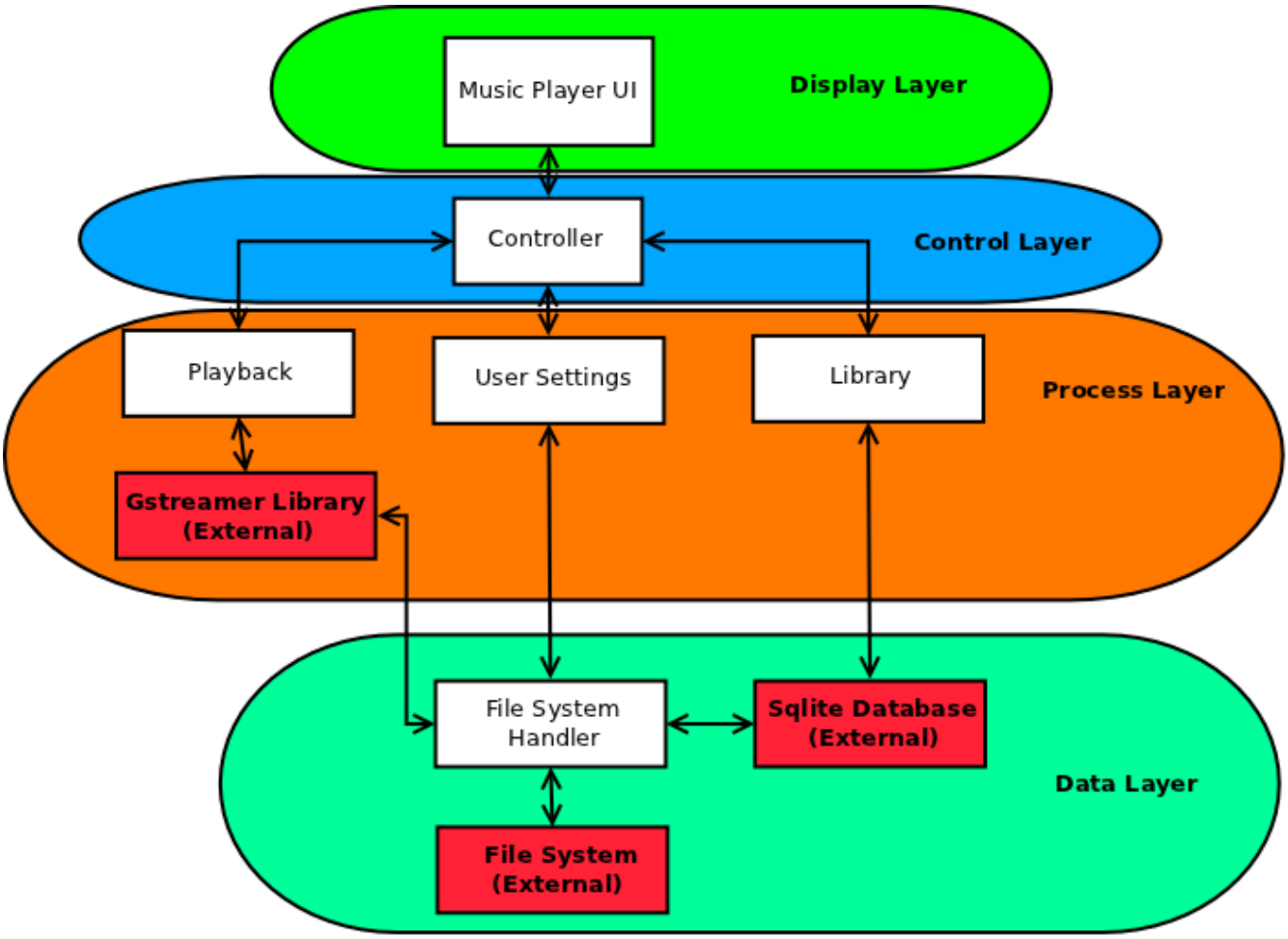
**Comment all of your class files for your project and compress them. You will have in this compressed file as part of the lab write up.**

| R · I · T | Rochester Institute of Technology<br>Golisano College of Computing and Information Sciences<br>Department of Information Sciences and Technologies |
|---|---|

**Overview – Architecture Documentation**

An important part of any software is the code design, architecture and the documentation. Architecture diagrams are important in highlighting how the components interact. There are a number of standards for this, but we will just use a simplified approach for now.

Architecture Example:



| R · I · T | Rochester Institute of Technology<br>Golisano College of Computing and Information Sciences<br>Department of Information Sciences and Technologies |
|---|---|

**Activity – Architecture Documentation**

Create an architecture diagram for your project. You can decide the names of any layers you want (your project may not have layers too). Just make sure you show the flow of your components. Any external libraries/packages you use can just be listed by their name. You do not need to convey their architecture.

**You will hand this in as part of the lab write up.**