# Assignment 2 : Regular expression/ text processing tools/ autotools

Emil Sharifulllin, Innopolis University
September 25, 2016

## 1 Assignment 1

### 1.1 Debian packaging system

The main Debian package management tool is **dpkg**. Dpkg is a program that allows users to install remove and configure packages. dpkg needs *.deb files which is special archives, containing binary executables, configuration files, man/info pages, copyright information, and other documentation. Debian systems also contain **apt**-Advanced Packaging Tool is a high-level tool that uses dpkg under itself. APT have repositories of .deb packages and there is not need to download .deb files manually.

#### 1.1.1 How does it work?

dpkg checks package for dependencies and if system doesn't contain certain dependency dpkg stops installation with error. Then it extracts files from .deb file, run preistall scripts, move binaries to system folder, run postinstall script. Last step is to add record to it's own registry, that in this system installed certain package.

apt is a tool that can automatically download .deb file, download dependencies .deb files. Then apt install package using dpkg.

#### 1.1.2 How does it deal with dependencies?

dpkg cannot install missing dependencies if dependency is not installed on system dpkg fails with installation. apt downloads and installs dependencies before install package. dpkg supports following types of dependencies:

- **A depends on B** B must be installed in order for A to function
- **A recommends B** A will work without B, but some useful functionality won't be available
- **A suggests B** a weaker version of recommends, that is, A is "nice to have" together with B
- **A conflicts with B** A and B cannot exist on the same system
- **A replaces B** B files will be overwritten by A's files
- **A breaks B** A will render B unusable

- **A provides B** all functions of B are supplied by A
- **A enhances B** reverse variant of suggests

#### 1.1.3 Does it use the GNU build tools? How? If not what does it use?

dpkg don't use build tools to install package because packages spreads as a .deb files which contains already built binaries. But dpkg uses build tools to create deb files for example in dpkg-buildpackage script.

## 1.2 Open SUSE packaging system

Open SUSE system contains RPM and zypper package systems. RPM is very similar to dpkg and works with compiled binary executables. RPM can install, deinstall and update packages. zypper is similar to apt and contain repository and can automative download and install packages.

#### 1.2.1 How does it work?

RPM works with .rpm files which is binary files and contains following data:

- The lead, which identifies the file as an RPM file and contains some obsolete headers.
- The signature, which can be used to ensure integrity and/or authenticity.
- The header, which contains metadata including package name, version, architecture, file list, etc.
- A file archive (the payload), which usually is in cpio format, compressed with gzip. The rpm2cpio tool enables retrieval of the cpio file without needing to install the RPM package.

To create new rpm package usually uses SPEC file. This is special recipe file that contain data about package and steps to compile package.

#### 1.2.2 How does it deal with dependencies?

RPM does not care about dependencies. To handle dependencies you should use high level package managers for example zypper or YaST. Zypper have own repository so it can download dependent packages from it and preinstall it.

#### 1.2.3 Does it use the GNU build tools? How?

As a dpkg RPM works with pre-builded binary packages and build tools is used during cretion of package.

## 2 Assignment 2

### 2.1 Look for a small and simple game on the web

For this assignment I chose TROG game:

#### 2.1.1 Download its source package for Ubuntu

Source code for TROG game can be fuod at https://github.com/JohnAnthony/TROG.git git repository.

#### 2.1.2 Create a binary from the source for the desktop

To create binary and install game on my system I heed to make following commands:

```
$ git clone https://github.com/JohnAnthony/TROG.git
$ cd TROG
$ sudo apt install ncurses-dev -y
$ make
$ sudo make install
```

My system by default not contain ncurses library so I need to install it manually.

#### 2.1.3 Read about cross-compilation and try to create a binary from the source for the i386 architecture. Try and install that on your experimentation computer.

Game that I downloaded didn't have configure script. So to compile it on my system I needed to change Makefile.

```
CFLAGS+= -std=c++0x -Wall -pedantic -O0 -ggdb -Iinclude -m32
OBJECTS= spell.o trog.o level.o game.o character.o enemy.o geometry.o tile.o \
item.o gui.o potion.o scrollable_menu.o treasure.o stattome.o equippable.o \
debug.o enemy_type.o
LIBS= -lncurses
CXX ?= g++

trog: ${OBJECTS}
	${CXX} ${CFLAGS} -o $@ ${OBJECTS} ${LIBS}

#Objects with a header file
%.o: src/%.cpp include/*.hpp
	${CXX} ${CFLAGS} -c -o $@ $<

clean:
	-rm -f $(OBJECTS) trog

install: trog
	cp -v trog /usr/bin/

.PHONY: clean
```

In this file I added -m32 option to CFLAGS variable.

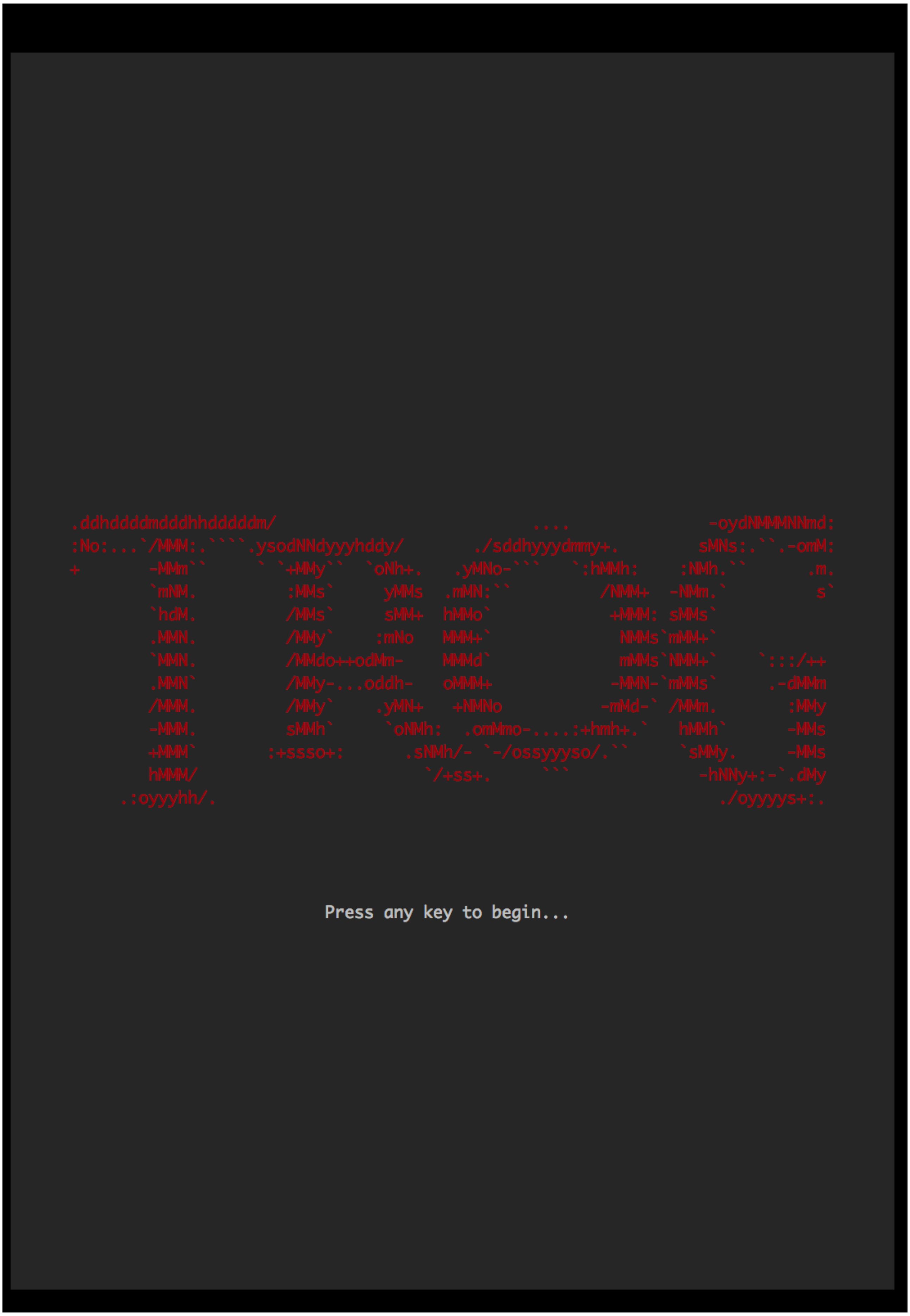
To allow gcc compile this code in crosscompile mode I installed following libraries.

```
$ sudo apt install gcc-multilib g++-multilib -y
$ sudo apt install lib32ncurses5-dev -y
```

After it I make following:

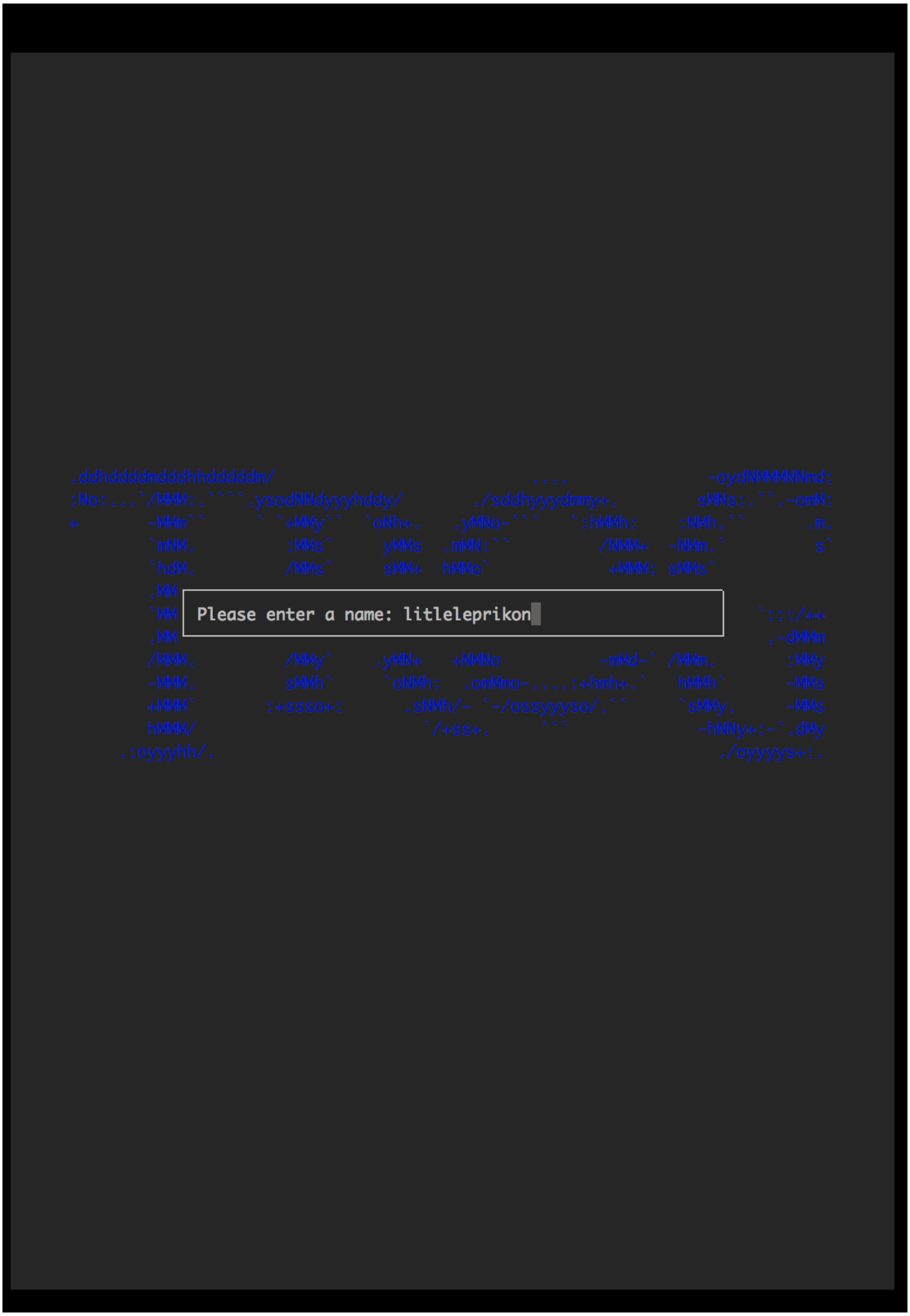
```
$ make clear
$ make
$ file trog
trog: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter
 ↪  /lib/ld-linux.so.2, for GNU/Linux 2.6.32,
 ↪  BuildID[sha1]=b413be74881edbcd7c1ff3f3b9fb294d15f8610e, not stripped
```

Final result:

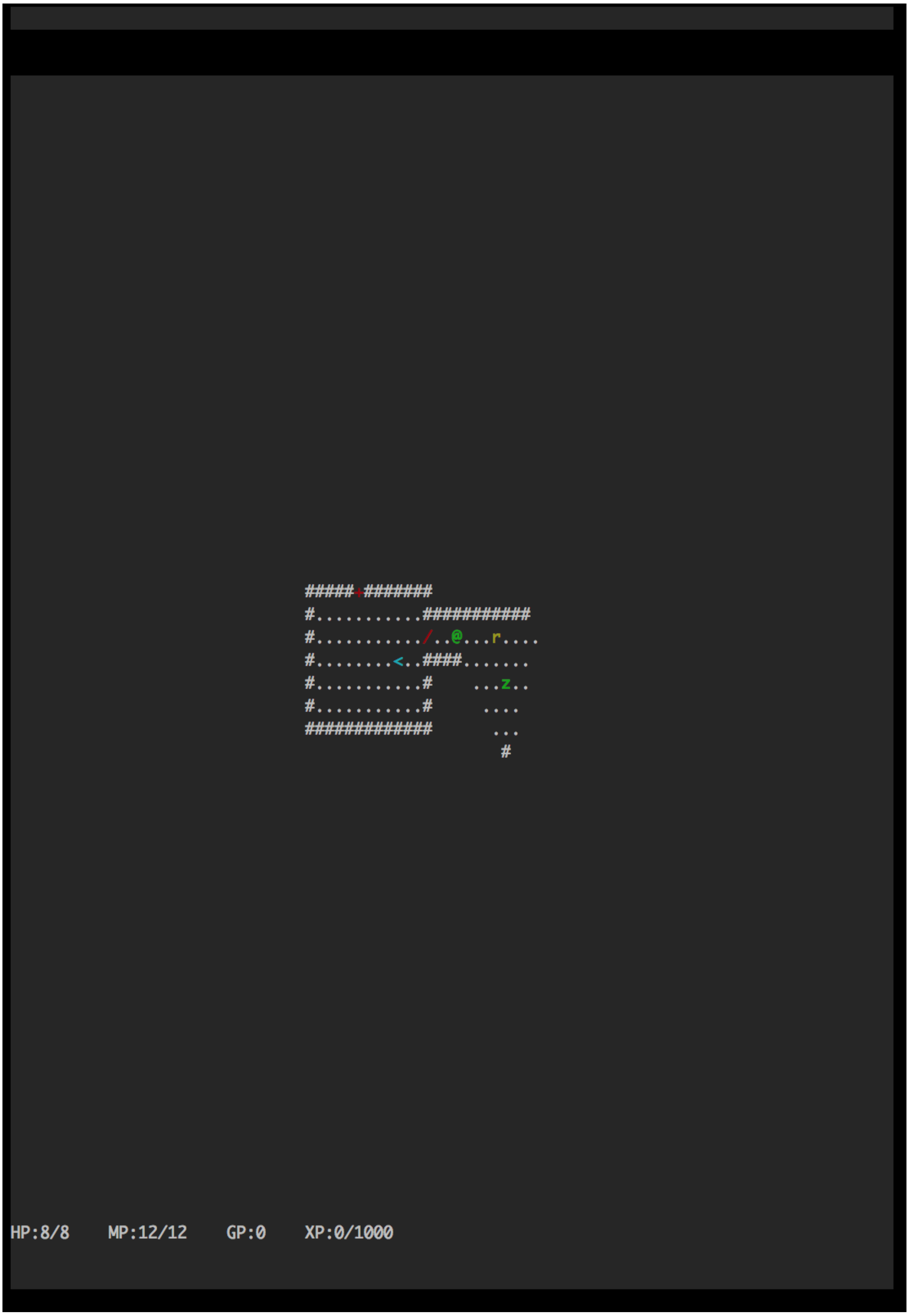


```
.ddhddddmdddhhdddddm/                          ....              -oydNMMMNNmd:
:No:...`/MMM:.````.ysodNNdyyyhddy/       ./sddhyyydmmy+.        sMNs:.``.-omM:
+       -MMm``     ` `+MMy``  `oNh+.   .yMNo-```   `:hMMh:    :NMh.``      .m.
        `mNM.         :MMs`     yMMs  .mMN:``         /NMM+   -NMm.`        s`
        `hdM.         /MMs`     sMM+  hMMo`            +MMM: sMMs`
        .MMN.         /MMy`    :mNo   MMM+`             NMMs`mMM+`
        `MMN.         /MMdo++odMm-    MMMd`             mMMs`NMM+`    `:::/++
        .MMN`         /MMy-...oddh-   oMMM+            -MMN-`mMMs`     .-dMMm
        /MMM.         /MMy`    .yMN+   +NMNo          -mMd-` /MMm.       :MMy
        -MMM.         sMMh`     `oNMh:  .omMmo-....:+hmh+.`   hMMh`      -MMs
        +MMM`       :+ssso+:      .sNMh/- `-/ossyyyso/.``     `sMMy.     -MMs
        hMMM/                       `/+ss+.     ```             -hNNy+:-`.dMy
     .:oyyyhh/.                                                   ./oyyyys+:.


                            Press any key to begin...
```



```
.ddhddddmdddhhdddddm/                          ....              -oydNMMMNNmd:
:No:...`/MMM:.````.ysodNNdyyyhddy/       ./sddhyyydmmy+.        sMNs:.``.-omM:
+       -MMm``     ` `+MMy``  `oNh+.   .yMNo-```   `:hMMh:    :NMh.``      .m.
        `mNM.        :MMs`     yMMs  .mMN:``         /NMM+  -NMm.`         s`
        `hdM.        /MMs`     sMM+  hMMo`            +MMM: sMMs`
        .MM
        `MM  Please enter a name: litleleprikon                       `:::/++
        .MM                                                            .-dMMm
        /MMM.        /MMy`    .yMN+   +NMNo          -mMd-` /MMm.       :MMy
        -MMM.        sMMh`     `oNMh:  .omMmo-....:+hmh+.`   hMMh`      -MMs
        +MMM`       :+ssso+:      .sNMh/- `-/ossyyyso/.``     `sMMy.     -MMs
        hMMM/                       `/+ss+.     ```             -hNNy+:-`.dMy
     .:oyyyhh/.                                                   ./oyyyys+:.
```



```
#####+#######
#...........###########
#.........../..@...r....
#........<..####.......
#...........#    ...z..
#...........#     ....
#############      ...
                    #
```

```
HP:8/8    MP:12/12    GP:0    XP:0/1000
```

## 3 Useful links

- Basics of the Debian package management system
- RPM on wikipedia
- askubuntu.com - gcc won't link with -m32