Dev + GNU Pascal Readme

This is a distribution of GNU Pascal (GPC) bundled with a significantly modified version of the *DevPascal* IDE (hence the name "Dev+GNU Pascal"). It has started to develop quite differently from the original *DevPascal*, and those who are used to that version should note that there are now many differences (some of them very subtle) between it and this distribution.

GPC can be run by itself from the command line, or via the Dev+GNU Pascal IDE.

Please read these notes very carefully before attempting to use Dev+GNU Pascal or GNU Pascal (GPC).

First use

When you run the Dev+GNU Pascal IDE for the first time, a number of source files will be loaded automatically. One of them is called *build_units.pas*. This program is only there for the purpose of building the units that come with this distribution, since it USES most of the units that are available with this distribution.

You should first build this program by selecting "**Build all**" from the "**Compile**" menu.

Building this program for the first time will most likely take quite a bit of time (depending on your CPU's speed). The compiled program must not be executed. If you try to run it, it will either hang, or strange things will happen with your printer, or both. Simply delete build_units.exe once it has been built.

By default, all compiled units are in the `<root>\units` directory. After building *build_units*, each "*make*" or "*compile*" should take far less time.

Extra "units"

This distribution includes Borland Pascal-compatible units (*Objects*, *Wintypes*, *Winprocs*, *Windows*), and Delphi-compatible units (*Sysutils*, *Messages*, *Windows*). They are in the <root>\units\ directory tree. You will also find there GPC ports of the *Graph* unit (and *Grx* unit) and a GPC port of the FreePascal *Mouse* unit (untested).

Also under that "units" tree is a free beta of my embryonic class library, called "*ObjectMingw*" (somewhat similar to the old ***OWL*** framework). When this class library is finished, it *may* become *shareware*. There are a number of sample programs that show how to use the class library. There isn't much else by way of documentation – so you will have to rely on the source code and the sample programs. All comments, suggestions, and bug reports are fully welcome.

The Dev+GNU Pascal IDE

GPC is a command line compiler. However, this distribution comes with an Integrated Development Enviroment (IDE) called Dev+GNU Pascal. Dev+GNU Pascal was originally written for the Freepascal compiler. I have extended it to support GPC as well as Freepascal. If you need an IDE, then the Dev+GNU Pascal IDE will do the job. It is still pretty much work in progress, and there may be bugs. Please report any problems.

The Dev+GNU Pascal IDE (*devpas.exe*) is in the main installation directory (i.e., the directory referred to as "<root>" in this document).

If you want to use Dev+GNU Pascal, you should first configure the IDE under the "Options/Compiler options" menu.

CUSTOM DIRECTIVES IN THE IDE

The Dev+GNU Pascal IDE recognises certain directives that the GPC compiler does not necessarily understand. These directives must be inserted very near the top of the source file in which they are used. Their mere presence in the first 4kb or so of the source file will trigger them in the IDE (i.e., the IDE just does a quick surface scan of the first 4kb of the source file, and no extra sophisticated processing - e.g., of conditionals - is done).

So for example, if you want to build a console application (which is actually the default GPC compiler setting), you can embed any of these at the beginning of the source file:

```
{$apptype console}
{$console}
{#apptype console}
{#console}
{#cconsole}
```

If you want to build a GUI application you can embed any of these at the beginning of the source file:

```
{$gui}
{$apptype gui}
{#apptype gui}
{#gui}
{#cgui}
```

If don't want any linking to be done (e.g., as with the build_units.pas program), then you can use either of these:

```
{$no_link}
{#no_link}
```

If you want the program that you are writing to always be rebuilt from scratch (i.e., build the program and all the supporting units) then use either of these:

```
{$auto-build}
{#auto-build}
```

If you are used to "no-fuss" compilers (i.e., brain-dead ones that take liberties with the programmer's intentions) and wish to just build your program with a minimum of fuss and specific configuration choices, then use either of these:

```
{$fast-mode}
{#fast-mode}
```

This is a "*quick and dirty*" mode that passes the following arguments to the GPC compiler:
"`--automake --implicit-result --extended-syntax --ignore-function-results --cstrings-as-strings --pointer-arithmetic`"

NOTE: the directives starting with "$" might well generate a warning from the GPC compiler that the directive is unknown. If you do not want any warnings, then use the "#" directives instead.

Code Completion

The Dev+GNU Pascal IDE supports code completion. This is activated by pressing Ctrl-SPACE when editing a source file. This provides a list of codes which can be inserted into the source file. The codes can be edited by going into the “Options/Environment Options/Code Completion” menu. From version 1.94.2 onwards, the entries on the list can contain a number of (case-sensitive) embedded formatting codes, which are translated when the codes are inserted into the source file. This is a list of the formatting codes and their meanings;

```
\r = Carriage return
\n = New line (Carriage return + Line feed)
\t = Tab
\s = Space
```

For example, this line (the formatting codes are in red in this example):

```
procedure foo;\nbegin\n\nend;{procedure}\n
```

translates into this, in the source file being edited:

```
procedure foo;
begin

end; {procedure}
```

LICENCE

Dev+GNU Pascal is released under the GNU General Public Licence. The original source code is available at the Bloodshed Software web site. The source code to this particular distribution is available at the same place that you got this distribution (assuming that you got it from the GNU Pascal website).

The GPCC extra program

This distribution of GPC comes with an extra program called “GPCC” for command line work. The program (“*gpcc.exe*”) will be found in your “bin” directory, alongside the GPC compiler driver (“*gpc.exe*”). GPCC (“GNU Pascal Compatibility Compiler”) is a command line shell for the GPC compiler. It attempts to mimic some of the features of Borland’s Pascal command line compilers (*tpc*, *bpc*, and *dcc32*). Those who are familiar with these compilers will find it easy to use GPCC and its configuration file (“*gpcc.cfg*”), as will those who are new to GPC. The *gpcc.cfg* file that is supplied should be sufficient for most needs. To use GPCC, you just need to run:

```
  gpcc <source_file[.pas]>
```

You can of course supply switches at the command line as well. More details can be found in *<root>\doc\gpc\gpcc.txt*.

GPC CONFIGURATION

The installation program has created a batch file called “*setgpc.bat*” in the installation directory. It gives an example of how you can configure the command line environment for GPC. If you prefer to run GPC from the command line rather than to use the Dev+GNU Pascal IDE, then open a command prompt, run “*setgpc.bat*”, and then you can, from the same command prompt, run “*gpc*” or “*gpcc*” to compile your programs.

EXAMPLE PROGRAMS

There are some example programs in these directories:

```
   <root>\examples\
   <root>\units\objmingw\
```

Some of them are loaded automatically into the IDE the first time that you run it.

THE MINGW ENVIRONMENT

This distribution of GPC for Mingw comes with everything that you need to build Pascal programs. However, if you want to build the runtime units yourself, then you might need to download and install the full Mingw distribution yourself (from http://www.mingw.org).

If you don’t have the GNU debugger (*gdb*) installed, then you will need to download it from the Mingw download site and install it if you want to be able to debug your programs.

Note: *gdb* is a text mode debugger, and is probably not for the fainthearted. However, it beats trying to locate the cause of a crash manually.

September 2007, Professor Abimbola A Olowofoyeku (The African Chief)
http://www.greatchief.plus.com