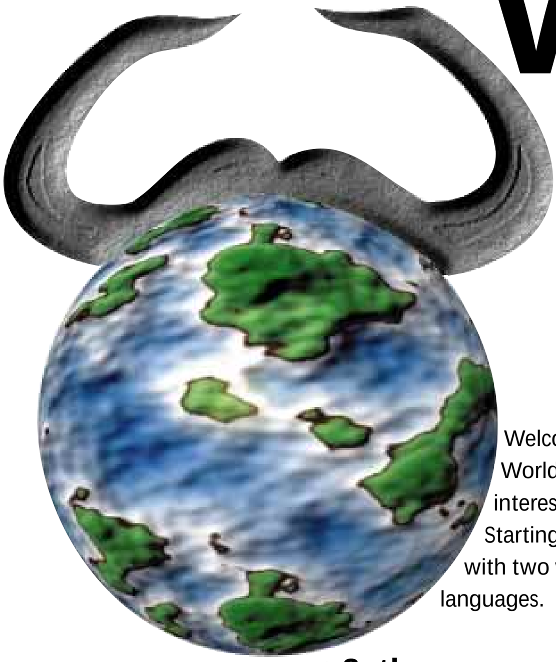
The monthly GNU-column BRAVE GNU WORLD



Georg C. F. Grave reports on the current developments and progress within the GNU-Project and tries to explain its philosephy. In this issue, you can read about GNU Sather, Ruby, a386, Guppi.

Welcome to Georg's Brave GNU World. I hope to have found an interesting mix of topics this month. Starting in the heart of technology with two very interesting programming languages.

Sather

GNU Sather [5] is an object oriented programming language that was originally branched off from Eiffel, but it has been changed in so many ways since that it must be regarded as an independent language.

The beginning of GNU Sather was as a scientific project at the ICSI, Berkeley, where it was distributed under a license that did not quite qualify as a Free Software license. But after development was stopped for financial reasons in 1998 a bunch of people were able to convince the officials at the ICSI to release the last version under the GPL/LGPL. This made it possible for GNU Sather to become an official GNU Project. Among the remarkable things about GNU Sather is its revolutionary interface concept where class interfaces are completely separated from their implementations; this makes multiple inheritance very easy. It is also possible to change the underlying code completely without touching the interface, should it become necessary.

The current maintainer of GNU Sather, Norbert Nemec, highlights the iterator concept, which allows all kind of loop concepts that other languages use to be implemented with a single break-statement. His view is also that GNU Sather is not just "another design study" - it is a language that has been designed for speed and comfort for the developer right from the start.

The current status of GNU Sather should probably be best described as "almost ready for day-to-day use." The interface to C and Fortran is easy and well-documented, so practically everything should be possible. The biggest weakness right now is the compiler which doesn't use the possible optimisations and has to be called "buggy." Obviously writing a new compiler is on top of the task list - but this will take some more time. The library also needs some more work, which is currently being done by the University of Waikato.

Despite these rough edges, developers interested in object oriented programming should check out GNU Sather. If nothing else it'll be an interesting

experience. Especially the integrated support for parallel computing (multi-threading up to TCP/IP clusters) and the library that was based on internationalisation since it's inception should make this an excellent tool once the latest problems have been solved.

Ruby

Ruby [6] by Yukihiro Matsumoto is another object-oriented programming language; it started in 1993 when the author wasn't able to find an object oriented scripting language and made the decision to write one himself.

The name Ruby was chosen because the author was looking for another "jewel name" to symbolize the closeness to Perl. His declared goal is to make Ruby the successor to and replace Perl. To achieve this he took the strengths of languages like Perl, Python, Lisp and Smalltalk and tried to incorporate these into Ruby.

Just like Perl, Ruby is very good at text processing and additionally gains from it's very broad object orientation. All data in Ruby is an object - there are no exceptions. The number "1," for instance, is an instance of the "Fixnum" class. It is possible to add methods to a class at runtime - even to an instance if need be. These possibilities make Ruby very flexible and extensiive. Additionally, it supports iterators, exceptions, operator overloading, garbage collection and much more that one likes to see in a language. In order to be able to replace Perl, it is also very portable and runs under GNU/Linux (and other Unices) as well as DOS, MS Windows and Mac.

Ruby has CGI-classes that allow easy CGI programming, and modules for the Apache also exist: eRuby (embedded Ruby) and mod_ruby. It contains a well thought-out network socket class, and thanks to Ruby/Tk and Ruby/Gtk it is possible to implement GUIs easily. There are also special features for the treatment of XML and an interface to the expat XML parser library.

Finally Ruby supports multithreading independently of the operating system - even under MS-DOS. Despite this complexity the syntax has been kept as simple as possible (inspired by Eiffel and Ada).

Ruby can be distributed under the GNU General Public License or a special license that gives users stronger "proprietarisation rights," but it might qualify as a Free Software license - although this remains to be thoroughly checked.

Although its features have probably mostly technical value, I do think that even non-programmers could be interested in developments in this area.

a386

a386 [7] is a library by Lars Brinkhoff that has a virtual Intel 386 CPU running in protected mode as a "Virtual Machine" (VM). This should be of most use to kernel hackers and scientists, but might also prove interesting for people just wanting to check out another operating system within their existing system.

Compared to similar projects like the Brown Simulator or plex86, a386 has the advantage of running privileged operations faster because they are implemented as function calls or inlined code. Additionally it aims for portability - working other CPU architectures as well as other operating systems.

Currently the task at hand is to enhance the Linux port, but in the medium term he seeks to create a NetBSD &amp; HURD port as well as making a386 run on these operating systems. The long term goal is to take the experience gained with a386 to create a new machine model which will be an abstraction of the wide-spread workstation/server CPUs and to implement this as a C library and a "Nano-Kernel" running directly on the hardware.

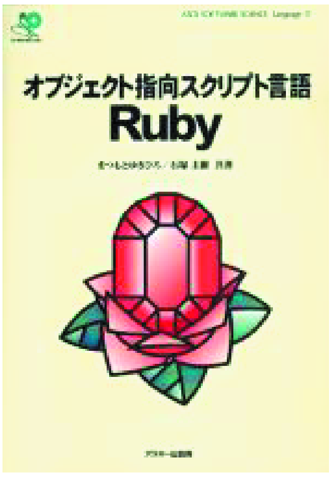
Of course everything is distributed under the GNU General Public License and if you're interested in these things, a look on the project's homepage might be a good idea [7].

But now I'd like to talk about things of more direct importance to the end-user.

Guppi

Strictly speaking, Guppi [8] is three things in one. First of all it is an application for data analysis and the creation of graphs and charts. Then it is a Bonobo component which allows embedding this functionality in other applications, and finally it's a set of libraries that allows any GNOME application to use it.

The application itself is definitely important for everyone relying on visualisation and analysis of empirical data - especially scientific users. In fact Guppi is the only program of its kind based on full GNOME integration from the start, and so it seems that it is slowly becoming the GNOME standard for visualisation. Thus, it is not surprising that the GNOME spreadsheet Gnumeric and the finance manager GnuCash rely on Guppi.



Info

[1] Send ideas, comments and questions to column@gnu.org
[2] Homepage of the GNU Project http://www.gnu.org/
[3] Homepage of Georg's Brave GNU World http://www.gnu.org/brave-gnu-world/
[4] "We run GNU" initiative http://www.gnu.org/brave-gnu-world/rungnu/rungnu.en.html
[5] GNU Sather home page http://www.gnu.org/software/sather
[6] Ruby home page http://www.ruby-lang.org/
[7] a386 home page http://a386.nocrew.org/
[8] Guppi home page http://www.gnome.org/guppi
[9] Jon Trowbridge trow@gnu.org

■

Gupy in action ...

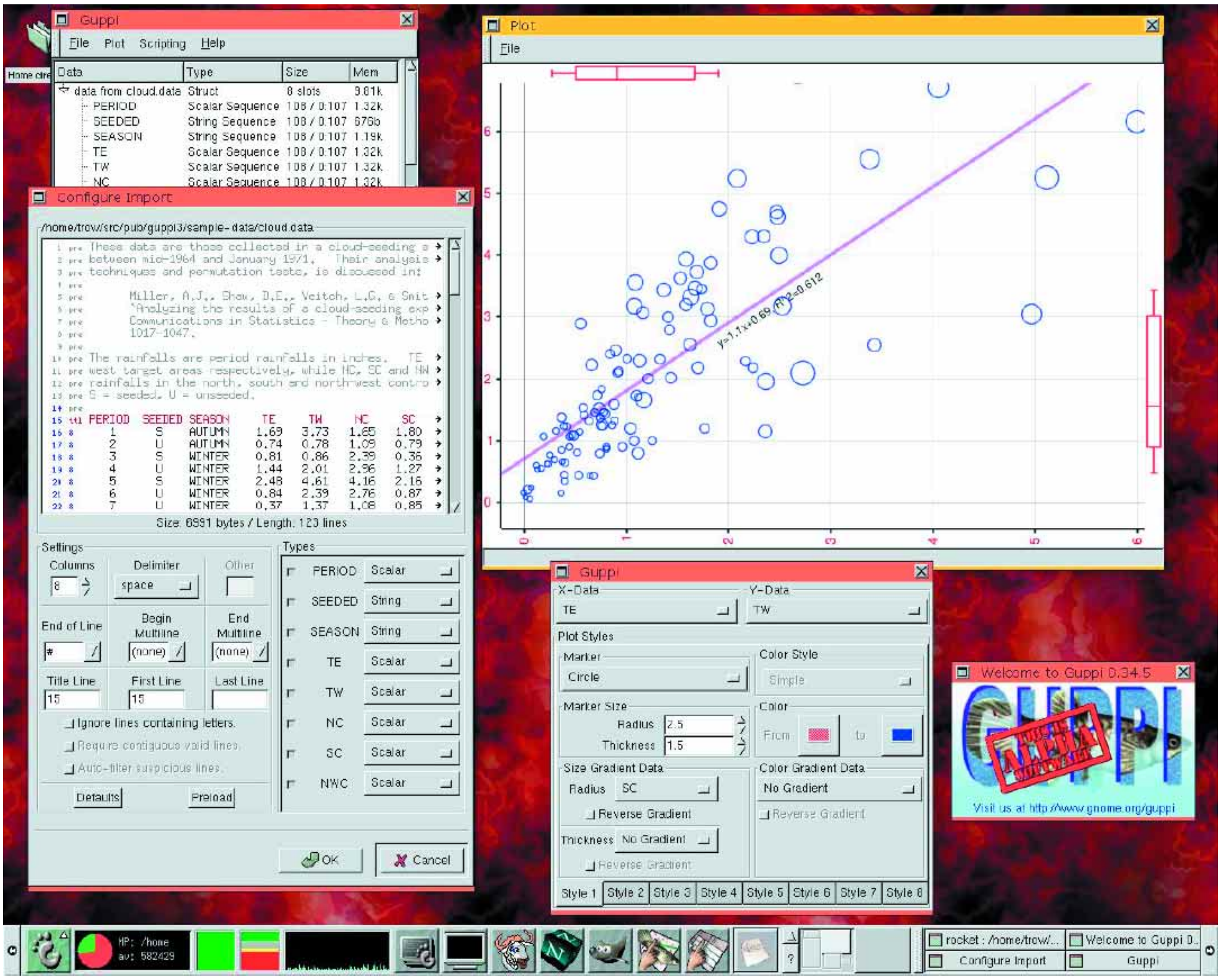
According to Jon Trowbridge, current maintainer of Guppi, the big advantages can be summed up in four points. First of all Guppi is scriptable, the internal API is available via Guile and Python - so it is possible to solve rather complex problems without having to program in C. Second Guppi has a very flexible data import filter with good guessing capabilities as to how a file should be read without intervention by the user. Third a lot of the functionality is broken down into plugins which makes it easy to extend, and finally Guppi has a WYSIWYG interface that should not give anyone trouble.

But the end-user should still be a little careful - right now Guppi is still in very active development and the user interface especially is not yet complete. There are also some functions lacking and the documentation is somewhere between sparse and non-existent, so only expert users should consider it for daily use.

Other members of the Guppi team are Jody Goldberg and Michael Meeks who work on the GNOME integration, Andrew Chatham, who takes care of the Python-binding, and I should also mention Havoc Pennington who doesn't work actively on Guppi anymore but did a majority of the work in the early phase. Anyone interested in development is very heartily invited to get in touch with Jon [9] - he also informed me that his current location is close to the University of Chicago (USA) and that he'd be interested in meeting more GNOMEs in this area.

...the end

Okay. That should be enough for this month… as usual I'm encouraging you to send your ideas, comments, questions and topic suggestions via email [1]. ■



Data	Type	Size	Mem
data from cloud.data	Struct	8 slots	9.81k
PERIOD	Scalar Sequence	108 / 0:107	1.32k
SEEDED	String Sequence	108 / 0:107	676b
SEASON	String Sequence	108 / 0:107	1.19k
TE	Scalar Sequence	108 / 0:107	1.32k
TW	Scalar Sequence	108 / 0:107	1.32k
NC	Scalar Sequence	108 / 0:107	1.32k

/home/trow/src/pub/guppi3/sample-data/cloud.data

```
pre These data are those collected in a cloud-seeding e
pre between mid-1964 and January 1971.   Their analysis
pre techniques and permutation tests, is discussed in:
pre
pre       Miller, A.J., Shaw, D.E., Veitch, L.G. & Smit
pre       "Analyzing the results of a cloud-seeding exp
pre       Communications in Statistics - Theory & Metho
pre       1017-1047.
pre
pre The rainfalls are period rainfalls in inches.   TE
pre west target areas respectively, while NC, SC and NW
pre rainfalls in the north, south and north-west contro
pre S = seeded, U = unseeded.
pre
ttl PERIOD  SEEDED SEASON    TE     TW     NC     SC
8      1       S   AUTUMN   1.69   3.73   1.65   1.80
8      2       U   AUTUMN   0.74   0.78   1.09   0.79
8      3       S   WINTER   0.81   0.86   2.39   0.36
8      4       U   WINTER   1.44   2.01   2.96   1.27
8      5       S   WINTER   2.48   4.61   4.16   2.16
8      6       U   WINTER   0.84   2.39   2.76   0.87
8      7       U   WINTER   0.37   1.37   1.08   0.85
```

Size: 6991 bytes / Length: 123 lines

Settings: Columns 8; Delimiter space; Other; End of Line #; Begin Multiline (none); End Multiline (none); Title Line 15; First Line 15; Last Line

Ignore lines containing letters.
Require contiguous valid lines.
Auto-filter suspicious lines.

Defaults Preload

Types	
PERIOD	Scalar
SEEDED	String
SEASON	String
TE	Scalar
TW	Scalar
NC	Scalar
SC	Scalar
NWC	Scalar

OK Cancel

X-Data: TE
Y-Data: TW
Plot Styles
Marker: Circle
Color Style: Simple
Marker Size: Radius 2.5, Thickness 1.5
Color: From 🟥 to 🟦
Size Gradient Data: Radius SC; Reverse Gradient; Thickness No Gradient; Reverse Gradient
Color Gradient Data: No Gradient; Reverse Gradient
Style 1 Style 2 Style 3 Style 4 Style 5 Style 6 Style 7 Style 8