Reprinted from the

Proceedings of the Linux Symposium

July 23th–26th, 2003
Ottawa, Ontario
Canada

Gnumeric

Using GNOME to go up against MS Office

Jody Goldberg
jody@gnome.org

Abstract

MS Office is popular for a reason. Microsoft and its massive user base have kicked it hard, and polished the roughest edges along the way. The hidden gotcha is that MS now holds your data hostage. Their applications define if your data can be read, and how you can manipulate it. The Gnumeric project began as a way to ensure the GNOME platform could support the requirements of a major application. It has evolved into the core of a spreadsheet platform that we hope will grow past the limitations of MS Excel. Gnumeric has taught us a lot about spreadsheets and, for the purpose of this talk, about what types of capabilities MS has put into its libraries and applications to provide the UI that people are familiar with. I'd like to discuss the tools (available and unwritten) necessary to produce a competitor and eventually a replacement.

1 Introduction

History

- Aug 17 1997 GNOME Created
- July 2 1998 Gnumeric Created
- Dec 31 2001 version 1.0.0 released
- Aug ?? 2003 version 1.2.0 released

Miguel de Icaza began work on Gnumeric in July 1998 with the stated purpose of building a large, real world, application to validate the GNOME libraries. Without much background in spreadsheets he persevered, learning as he went, writing maintainable code, and testing such core libraries as gnome-canvas, gnome-print, libgsf, and Bonobo.

Status

At this writing Gnumeric now has basic support for 100% of the spreadsheet functions shipped with Excel and can write MS Office 95 through XP, and read all versions from Excel 2 upwards. It covers most major features, but is still lacking pivot tables, and conditional formats which are planned for the 1.3 release cycle.

2 Container File Format

The first step in interoperability with MS Office is to read and write its files. MS Office 95 was an epochal event for the suite. Before then each application had its own format. As of Office 95 it shares a single container format used to wrap each application's native representation. This is convenient for embedding because it allows a user to transfer the entire content of a document, including all of its components as a single file.

2.1 OLE2, The MS solution

The wrapper format used throughout MS office is called *OLE2*. That title actually encompasses the container and a sub-format used to store metadata. GNOME had no support for reading or writing either one. For Gnumeric Michael Meeks used earlier work in laola by Arturo Tena and Caolan McNamara, and scraps of documentation to create libole2. In the last few years the set of available documentation has improved greatly. Apache's POIFS project has generated some nice coherent write-ups for their Java implementation. Additionally, although the Chicago project has not produced any code, they googled extremely well, and collected enough documentation to illuminate the remaining corners.

OLE2 is quite literally a filesystem in a file. It has a directory tree for the offsets and file allocation tables to store the layout of the blocks. Unfortunately for libole2 it also has a meta file allocation table, which libole2 could not export correctly. As a result, after approximately 6.8 MB of data, the library and all of its derivatives produced invalid results. Given the new documentation, we're written libgsf (the GNOME Structured File library) to solve that. The new code has been quite stable, and now that libole2 has been deprecated it has made its way into koffice, abiword, and several other applications. Including potentially OOo via the WordPerfect library.

2.2 The Future

- Format should be easy to create and manipulate without special tools.
- Existing filemagic type sniffing should be able to ID the documents as documents, not their container type
- Support for 'filesystem in a file' structured files to facilitate embedding and split data from metadata.
- Space & time efficient for reading and generation
- Portable to as many platforms as feasible
- signing
- encryption
- embedded object handling
- possible dual/multiple version support
- meta data

Things we do not need

Transaction support is probably overkill. Given the general size of documents, it seems simpler to just generate the entire file each time it is saved.

In-place update (rewrite one element without touching the rest) is also probably unnecessary. This is useful for things like presentation programs with large data blobs (images, sound) and relatively little content, it is also conceivably useful in situations where an external app is editing just metadata. However, the implementation costs for these are high in comparison to available manpower.

At the time of this writing, OOo's container format is the leading contender. There are discussions at this time to potentially adopt the container format (not the underlying xml content).

3 File Format

3.1 XLS

Like a Russian doll, parsing the wrapper format is just the beginning. Within the OLE2 files are sub-files called Book or Workbook (depending on version) that are in the *BIFF* format with several different variants. There is actually some reasonably good documentation for this due to some MS greed (tm) and hard work on the part of the OOo xls filter team. BIFF is fairly similar in structure to earlier lotus formats, and thanks to microsoft padding one of its books (The Excel 97 Developers Kit) with their internal file format docs we know a fair amount about how things are structured. Contrary to the common Slashdot wisdom the hard part is not in knowing the broad details of the records. The devil is in the details, implementers need to understand *what* Microsoft means for each flag and have a strict superset of the corresponding application to avoid information being lost when round tripping data to a proprietary format.

An odd truth of open-source spreadsheets is that they generally interoperate better via xls than their native formats. E.g., Gnumeric can read OpenCalc, but not write it, and OpenCalc has no support for Gnumeric at all. This speaks volumes for the ubiquitous nature of the Microsoft formats. The resource expenditure to support xls fully limits the amount of development time for other formats.

Within the BIFF records is nested yet another format to store the details of the expressions, which is also reasonably documented.

3.2 Escher

A major change in Office97 was the addition of a shared drawing layer for all of the applications. This allows you to draw an org chart in Excel and paste it into Powerpoint. Its high level format was reasonably documented on the web until that was pulled a few years ago. This content is nested within BIFF records within OLE2 records. Escher is a format in the classic 'specification by implementation' genre. Although both OOo and Gnumeric have decent parsers for the records themselves, parsing the content is tricky. One rarely knows what attributes correspond to visible properties. Exporting escher is even more complex. Both Gnumeric and OOo appear to have adopted a monkey see, monkey do approach to work around Microsoft's reluctance to use 'conventional' values for flags requiring things like 0xAAAAFFFF for true for some of the boolean flags.

3.3 WMF/EMF

Users expect their word/clip art to appear faithfully. That content is usually stored using a set of drawing primitives in a series of MetaFile formats (Windows and Extended). The primitives are slightly higher level than a serialized set of X protocol requests. There in an existing rasterizer for wmf in libwmf, but it can not currently handle emf. OOo has a parser for both wmf and emf, but it is tightly coupled to the OO platform and of marginal quality. Proper handling of this is still an open question. There have been discussions with the maintaers of libwmf, and libEMF to combine efforts to fill this niche, but nothing concrete has materialized as yet.

3.4 Security

Unfortunately each element of MS Office handles this slightly differently, and approaches vary from version to version. There is no secure notion of authentication within OLE2, or xls. Microsoft apparently assumes that is handled at a higher level. Within xls there are 3 main forms of encryption. The first is sheet level protection that is little more than an XOR of the records with a 16-byte hash of the password. This is tissue-thin, and can be supported trivially. Workbook level encryption is significantly more secure and uses md5 hashed passwords and rc4 to encrypt the BIFF record content. Workbook level protection is rather strange. It uses the secure workbook level encryption, with a hard coded password. Gnumeric is the only open source application that can handle all three.

3.5 VBA

This is still largely uncharted territory for open source applications. MS Office appears to store the VBA code in at least 3 formats.

1. Compressed source code. libgsf, libole2, and openoffice can all decompress the code with varying degrees of accuracy. What none of us knows is how to locate the offset to the start of the compressed stream. OO and libole2 both have kludges in place to guess, but neither is reliable. There is clearly documentation on the subject available to the anti-virus manufacturers, but its licensing precludes its use in open source libraries. This is the most likely route to support importing VBA in the near term. It is not immediately obvious that source code is what we really want, because it requires a lexer, parser, and libraries to back it up—a rather significant amount of work. There is some hope that the Mono project and its emulation of the .Net API will provide support for this.
2. P-Code. A preparsed set of tokens to be interpreted by the VB engine. The format of this has no open documentation, although it is fairly amenable to parsing. On the positive side this is the holy grail in many ways. Having pre-parsed code removes the need for a lexer and parser, and allows us to map the content to more modern languages such as python. The down side is that the p-code comes in many variants, and depends on versioning.
3. S-Code. Like P-Code, but different in some unspecified way. No known parsers or documentation exists.

4 Expansion

In addition to their core functionality, office applications are expandable. Organizations have some limited ability to customize their installations, and third party developers can use them as a development platform for their niche applications. From tasks as simple as workflow macros, up to massive extensions such as FEA's @Risk, or DeltaGraph, people are extending MS Excel.

4.1 XLLs & XLMs

Early versions of Excel offered 2 forms of extension. The first was a kludge that grafted a pseudo-procedural language into the functional format of a spreadsheet, called XLM. This is no longer widely used.

There is also the opportunity to load an XLL, a DLL shared library with special entry points, directly into the process' address space. Although its interface is only partially documented, somewhat byzantine, and deprecated, this is the most popular form of extension. The primary benefit is that a developer's code can be written in C/C++, and compiled to link the external libraries fairly easily.

To the best of my knowledge there are no open source or proprietary applications that support XLMs or XLLs other than MS Excel. This hinders transition. XLMs could potentially be supported, but the limited remaining user base does not warrant the expense. XLLs might be feasible under win32, but due to the nature of the interface, would probably require WINE under *nix.

4.2 VBA & OLE

With Office97 came a unification of embedding and scripting via VBA and the OLE component model. VBA as a language was a significant improvement over XLM and quickly supplanted it. OLE was more of a mixed blessing. The increased complexity of the interface required Dev Studio wizards to generate the wrapper code, which was fairly unmaintainble. As a result most installations fell back on VBA external declaration support to add new capabilities. This worked well for tasks amenable to scripting, but was painful when linking to external analytics. Adding a new spreadsheet function involved writing it in C, then creating a dummy wrapper in VBA that links to it.

4.3 Gnumeric

Worksheet function management is an area where Gnumeric is well ahead of MS Office. Adding new functions to Gnumeric is trivial. An abstract interface for loading modules has been implemented for shared libraries (via glib's g_module utilities), and python. Coupled with an xml-based configuration mechanism and just-in-time loading, the vast majority of the worksheet functions are in plugins.

Less well defined are the scripting interfaces. Building on GNOME's strong set of language bindings there have also been experiments in scripting in guile, perl, CORBA, VB, and python. The only clear result thus far has been that the scripting interface is fairly language-agnostic. Defining a clear and coherent api is on the short list of extensions to be made during the 1.3–to–2.0 development cycle.

5 Preferences

Storing user preferences is another area where GNOME technology has the advantage over its Windows counterpart. GConf attempts to learn the lessons of the Windows registry while learning from its failures. By storing content in several distinct user readable xml files, gconf offers the convenience of a global structured storage, while retaining the flexibility in the face of file errors or corruption not found in the more monolithic Windows registry. Work remains though. There is still some thought necessary to implement lockdown features, and to address logical paths (HOME, PREFIX, etc.).

6 GUI Toolkit

Over time, the initial separation between gnome libraries as extensions to gtk have been largely removed. With the addition of pango to handle advanced text, and extensions to Gtk's rendering model that produced the foocanvas, gtk+ now supports the primary display needs of Gnumeric. Coupled with libglade for easy maintenance and configurability, it is relatively painless to produce extremely usable dialogs. There are, however, a few lingering issues to address.

Configurable UI

The current gtk+ api for menus and toolbars makes no distinction between the actions and their layout. Applications are forced to hardcode their menu/toolbar layouts in order to modify them. This removes the ability of a user to reorganize things. The limitations of the gtk+ path-based API prompted the creation of GNOME_UI app helpers, which simplified creation and added stock items to improve consistency between GNOME applications. However, it did nothing to solve the issue of hardcoded layouts. In an effort to solve the problem of merging menus and toolbars for components, Bonobo made an attempt to solve the layout problems by separating the layout from the actions. Unfortunately, the API was produced without enough review, and it was insufficent for large applications. The hopefully final rendition is now in its evaluation phase in libegg menu/toolbar. This code allows effective management of different action groups, and the creation of new action types such as combos and accelerators. The main remaining question is how to store a user's edits. KDE has long had support for this sort of editing, they don't appear to have a good solution to storing the edits as yet.

File Selector

The gtk+ file selector has long been a source of ridicule and disgust. It is functional, but too barebones for a modern desktop. The fact that it has no solid support for network addresses, or histories has greatly hindered the adoption of gnome-vfs. There have been several write-ups and Owen Taylor has apparently completed a replacement version that will be included in gtk+-2.4.

7 Spreadsheet-specific Functionality

Spreadsheets are an ideal testing ground for all those obscure datastructures we all learned back in school. In most instances there is not enough data to make using something esoteric worthwhile. With an apparent size of 256 $\times$ 64k (Gnumeric can scale considerably larger), it is very easy to quickly operate on significant swaths of data.

As an example, Gnumeric uses an asymetric quadtree to store style information. This allows us to easily handle someone doing a "Select all, Bold" (explode kspread). It also supports "Select all minus one row and one col" (explode MS Excel, and OpenCalc).

8 Acknowledgements

The Gnumeric development team. You know who you are, as do the CVS logs.

Ximian employees for their continuing personal contributions to Gnumeric.

9 References

`http://www.gnome.org/projects/gnumeric`