O'Reilly Network for AvantGo

What Is P2P ... And What Isn't (P2P DevCenter)
P2P is not about computers acting as peers, writes Clay Shirky. It's about peers operating on the edge of the Internet, a place where IP addresses are transitory and connectivity is intermittent. [11/24/2000]

What's New in Qt 2.2.2
The popular cross-platform C++ toolkit is sporting hot new features and additional modules -- plus a preview of the Qt Palmtop Environment. [11/24/2000]

XML-RPC in Python (Python DevCenter)
Take advantage of this lightweight method for using services on other computers with Python's xmlrpclib module. [11/22/2000]

24 Hours With a Wireless Palm Vx (Wireless DevCenter)
Essential business tool or expensive toy for the mobile rich? Derrick Story puts a Palm Vx with a Novatel Minstrel V modem through its wireless paces. [11/22/2000]

Discovering System Processes Part II (BSD DevCenter)
Dru Lavigne takes us deeper into the realm of system processes and explains interprocess communication and signal handling. [11/22/2000]

What's on Freenet? (P2P DevCenter)
Jon Orwant takes a look at the content on Freenet and finds a bastion of sex, drugs, and rock 'n' roll. [11/21/2000]

The O'Reilly Network channel is under construction - please email help@oreillynet.com regarding problems or suggestions.

Published on The O'Reilly Network (http://www.oreillynet.com/)
http://www.oreillynet.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html

What Is P2P... And What Isn't

by Clay Shirky
11/24/2000

After a year or so of attempting to describe the revolution in file sharing and related technologies, we have finally settled on a label for what's happening: peer-to-peer.

Somehow, though, this label hasn't clarified things. Taken literally, servers talking to one another are peer-to-peer. The game Doom is peer-to-peer. There are even people applying the label to e-mail and telephones. Meanwhile, Napster, which jump-started the conversation, is not peer-to-peer in the strictest sense, because it uses a centralized server to store pointers and resolve addresses.

If we treat peer-to-peer as a literal definition for what's happening, then we have a phrase that describes Doom but not Napster, and suggests that Alexander Graham Bell was a peer-to-peer engineer but Shawn Fanning is not.

This literal approach to peer-to-peer is plainly not helping us understand what makes P2P important. Merely having computers act as peers on the Internet is hardly novel, so the fact of peer-to-peer architecture can't be the explanation for the recent changes in Internet use.

What *has* changed is what the nodes of these P2P systems are -- Internet-connected PCs, which had been formerly relegated to being nothing but clients -- and where these nodes are -- at the edges of the Internet, cut off from the DNS system because they have no fixed IP address.

Related Articles:

What's On Freenet?

Free Radical: Ian Clark has Big Plans for the Internet

How Ray Ozzie Got His Groove Back

A Directory of Peer-to-Peer Projects

Open Source Roundtable: Free Riding on Gnutella

O'Reilly's Peer-to-Peer Conference

More from the P2P DevCenter

Previous Features

Resource-centric addressing for unstable environments

P2P is a class of applications that takes advantage of resources -- storage, cycles, content, human presence -- available at the edges of the Internet. Because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses, P2P nodes must operate outside the DNS system and have significant or total autonomy from central servers.

That's it. That's what makes P2P distinctive.

Napster and ICQ and Freenet and AIMster and Popular Power and Groove are all leveraging previously unused resources, by tolerating and even working with the variable connectivity of the hundreds of millions of devices that have been connected to the edges of the Internet in the last few years.

P2P is as P2P does

Up until 1994, the whole Internet had one model of connectivity. Machines were assumed to be always on, always connected, and assigned permanent IP addresses. The DNS system was designed for this environment, where a change in IP address was assumed to be abnormal and rare, and could take days to propagate through the system.

With the invention of Mosaic, another model began to spread. To run a Web browser, a PC needed to be connected to the Internet over a modem. This created a second class of connectivity, because PCs would enter and leave the network cloud frequently and unpredictably, and would have a different, possibly masked, IP address with each new session. This instability prevented PCs from having DNS entries and therefore prevented PC users from hosting any data or Net-facing applications locally.

For a few years, treating PCs as dumb but expensive clients worked well. PCs had never been designed to be part of the fabric of the Internet, and in the early days of the Web, the toy hardware and operating systems of the average PC made it an adequate life-support system for a browser, but good for little else.

Over time, though, as hardware and software improved, the unused resources that existed behind this veil of second-class connectivity started to look like something worth getting at. At a conservative estimate, the world's Net-connected PCs presently host an aggregate ten billion Mhz of processing power and ten thousand terabytes of storage, assuming only 100 million PCs among the net's 300 million users, and only a 100 Mhz chip and 100 Mb drive on the average PC.

by Clay Shirky

The veil is drawn back

The launch of ICQ in 1996 marked the first time those intermittantly connected PCs became directly addressable by average users. Faced with the challenge of establishing portable presence, ICQ bypassed DNS in favor of creating its own directory of protocol-specific addresses that could update IP addresses in real time, a trick followed by Groove, Napster, and NetMeeting as well. (Not all P2P

systems use this trick. Gnutella and Freenet, for example, bypass DNS the old-fashioned way, by relying on numeric IP addresses. Popular Power and SETI@Home bypass it by giving the nodes scheduled times to contact fixed addresses, thus delivering their current IP address at the time of the connection.)

Whois counts 23 million domain names, built up in the 16 years since the inception of IP addresses in 1984. Napster alone has created more than 23 million non-DNS addresses in 16 months, and when you add in all the non-DNS Instant Messaging addresses, the number of P2P addresses designed to reach dynamic IPs tops 200 million. Even if you assume that the average DNS host has 10 additional addresses of the form foo.host.com, the total number of P2P addresses now equals the total number of DNS addresses after only 4 years, and is growing faster than the DNS universe today.

As new kinds of Net-connected devices like wireless PDAs and digital video recorders like TiVo and Replay proliferate, they will doubtless become an important part of the Internet as well, but for now PCs make up the enormous preponderance of these untapped resources. PCs are the dark matter of the Internet, and their underused resources are fueling P2P.

Litmus tests

If you're looking for a litmus test for P2P, this is it: 1) Does it treat variable connectivity and temporary network addresses as the norm, and 2) does it give the nodes at the edges of the network significant autonomy?

If the answer to both of those questions is yes, the application is P2P. If the answer to either question is no, it's not P2P.

Another way to examine this distinction is to think about ownership. It is less about "Can the nodes speak to one another?" and more about "Who owns the hardware that the service runs on?" The huge preponderance of the hardware that makes Yahoo work is owned by Yahoo and managed in Santa Clara. The huge proponderance of the hardware that makes Napster work is owned by Napster users and managed on tens of millions of individual desktops. P2P is a way of decentralizing not just features, but costs and administration as well.

Who's who?

Napster is P2P, because the addresses of Napster nodes bypass the DNS system, and because once the Napster server resolves the IP addresses of the PCs hosting a particular song, it shifts control of the file transfers to the nodes. Furthermore, the ability of the Napster nodes to host the songs without central intervention lets Napster users get access to several terabytes of storage and bandwidth at no additional cost.

However, Intel's "server peer-to-peer" is not P2P, because servers have always been peers. Their fixed IP addresses and permanent connections present no new problems, and calling what they already do "peer-to-peer" presents no new solutions.

ICQ and Jabber are P2P, because not only do they devolve connection management to the individual

nodes once they resolve the addresses, they violate the machine-centric worldview encoded in the DNS system. Your address has nothing to do with the DNS systems, or even with a particular machine, except temporarily -- your chat address travels with you. Furthermore, by mapping "presence" -- whether you are at your computer at any given moment in time -- chat turns the old idea of permanent connectivity and IP addresses on its head. Chat is an important protocol *because* of the transience of the connectivity.

E-mail, which treats variable connectivity as the norm, is nevertheless not P2P, because your address is not machine independent. If you drop AOL in favor of another ISP, your AOL e-mail address disappears as well, because it hangs off DNS. Interestingly, in the early days of the Internet, there was a suggestion to make the part of the e-mail address before the @ globally unique, linking e-mail to a person rather than to a person@machine. That would have been P2P in the current sense, but it was rejected in favor of a machine-centric view of the internet.

Popular Power is P2P, because the distributed clients that contact the server need no fixed IP address and have a high degree of autonomy in performing and reporting their calculations, and can even be offline for long stretches while still doing work for the Popular Power network.

Dynamic DNS is not P2P, because it tries to retrofit PCs into the traditional DNS system, and so on.

This list of resources that current P2P systems take advantage of -- storage, cycles, content, presence -- is not necessarily complete. If there were some application that needed 30,000 separate video cards, or microphones, or speakers, a P2P system could be designed that used those resources as well.

P2P is a horseless carriage

Whenever something new seems to be happening on the Internet, there is a push to define it, and as with the "horseless" carriage or the "compact" disc, new technologies are often labelled according to some simple difference from what came before -- horsedrawn carriages, non-compact records.

Calling this new class of applications peer-to-peer emphasizes their difference from the dominant client/server model. However, like the horselessness of the carriage or the compactness of the disc, the "peeriness" of P2P is more a label than a definition.

As we've learned from the history of the Internet, adoption is a better predictor of software longevity than perfection is, and as the P2P movement matures, users will not adopt applications that embrace decentralization for decentralization's sake. Instead, they will adopt those applications that use just enough decentralization, in just the right way, to create novel functions or improve existing ones.

Clay Shirky is a professor of new media at Hunter College, a columnist for the online magazine Feed and other publications, and a partner at the Accelerator Group, an incubator for Internet start-ups.

Related Articles:

What's On Freenet?

Free Radical: Ian Clark has Big Plans for the Internet

A Directory of Peer-to-Peer Projects

How Ray Ozzie Got His Groove Back

Open Source Roundtable: Free Riding on Gnutella 

O'Reilly's Peer-to-Peer Conference

Discuss this article in the O'Reilly Network General Forum.

Return to the P2P DevCenter.

Published on The O'Reilly Network (http://www.oreillynet.com/)
http://www.oreillynet.com/pub/a/network/2000/11/24/qt222_review.html

What's New in Qt 2.2.2?

by Boudewijn Rempt
11/24/2000

I just spent some time with the final release of Qt 2.2.2. Qt is best known as the toolkit underlying KDE, but it's also a very complete cross-platform C++ toolkit in itself, used by an increasing number of commercial applications such as Opera.

Qt applications can be run on three platforms: Unix/X11, Windows, and embedded Linux. Qt has been certified open source since version 2.0, the Unix/X11 version is released under GPL since version 2.2, and the embedded version is GPL since version 2.2.2. The Windows version remains proprietary in the sense that developers need to pay for a license, but there are no run-time restrictions. For instance, the GPLed Python bindings to Qt, PyQt, are available freely with a Qt dll for Windows.

Qt Designer is beautiful GUI

What's new in Qt 2.2? The coolest, most impressive addition is Qt Designer. This is a great GUI design utility. I've reviewed Qt Designer before (see Visual Design with Qt), but to sum it up: Qt Designer is a very stable, very easy-to-use dialog designer. It is especially good in creating complex dynamic layouts (helped by Qt 2.2's vastly improved layout managers) and in supporting the well-known signals/slots architecture.

The design is sensible: There's a separate GUI designer that saves the designs in an XML formatted file, and a compiler, uic, that takes the design files and creates C++ code. This opens up the possibility of writing all kinds of extensions to the system, and indeed, there is already a pyuic available that can generate Python code instead of C++. Designer is a truly wonderful tool, and now

The Qt Palmtop Environment

Palmtop is becoming an exciting area of Linux development. While most Linux PDA projects are still vaporware (see All Linux PDA: Fact or Fiction), other projects have come to full fruition. One of these is the Qt Palmtop Environment.

Qt Palmtop is licensed under the GPL. You can download the source these ways: a binary that runs on top of Linux, a binary that runs from a single boot floppy on any PC, a binary for the iPAQ, or a binary for the Cassiopeia.

The package is fairly large -- about 2.5 MB. Another project, Microwindows, has delivered a a package that needs a mere 100 KB for its environment, but that

with version 2.2.2, it can even import the Microsoft standard .RC dialog resources files.

OpenGL, XML modules, and more

With Qt 2.2, the toolkit has certainly become very comprehensive, offering almost everything that's needed for application development, and Troll Tech has decided to split the library into several modules. Paying customers can choose between a professional and an enterprise edition, the difference being the number of modules they receive. Users of the free edition get the full set, but their development has to be open source, too.

You can choose which modules are compiled into the library, but you cannot compile modules into smaller, separate libraries.

The following modules are available:

- Canvas - a sophisticated 2D drawing area that allows more than one concurrent view
- Iconview - an area with movable, labeled icons
- Network - a complete set of easy-to-use network classes
- OpenGL - a widget that allows rendering of OpenGL
- Table - an editable table widget
- Workspace - an MDI implementation
- XML - a SAX2 XML parser and a DOM tree implementation.

excludes the base OS and all applications.

Qt Palmtop is more than just a GUI and a window manager. It offers a choice of input methods (handwriting, picklists, Unicode) and all the utilities you need on a palmtop plus games and even an MPEG player!

Now you can comfortably run a Linux-based PDA and have it do everything that you want it to do. Hack it as much as you like! Qt Palmtop offers all the features of Qt for Windows or X11, such as the canvas, plus new features such as handwriting recognition.

Go to Troll Tech, and start playing...

XML parser and DOM implementation

The advent of Qt Designer has brought about several other new features. Most notable amongst these is the XML parser and DOM tree implementation. The parser has a SAX2-compliant interface, but the naming is adapted to the Qt standards. This fast XML parser has already been very useful in the development of the upcoming KDE 2 desktop.

Another improvement that should come as a godsend to developers is the QAction/QActionGroup combo. This is a mechanism whereby a certain user interface action can be defined at once both for a toolbar and a menu item, together with accelerator keys. These actions can then be logically grouped in QActionGroups. I've been building something like this time and again for my own applications ever since Qt 1.44, but never getting it really right, and the Troll Tech implementation works very well indeed.

Troll Tech has taken some flack for their QString class, with detractors arguing that Troll Tech should use the standard C library string. However, QString offers excellent Unicode support, and Troll Tech is committed to bringing the same level of Unicode support to Unix/X11 as Windows 2000 has. The

standard C library string just doesn't offer that.

And Qt offers a decent mechanism for producing localised applications, too. Version 2.2.2 brings numerous improvements to this area, with expanded support for Chinese and Thai. Internationalisation is further helped with the BSD-style licensed Qt Linguist.

Qt has now started building real support for threading, with QMutex, QSemaphore, QCondition, and QThread. Other extremely usable classes are available for networking applications, like QSocket, QServerSocket, or QNetworkOperation, and for encoding text, like QTextCodec, QTextEncoder, and QTextDecoder.

by Boudewijn Rempt

Properties -- useful or just added weight?

One relatively new feature that's heavily used by Qt Designer is the property concept. No longer is it necessary to use functions to get and set certain properties of widgets, such as font. Instead, we have a property mechanism that is implemented as a moc (Meta Object Compiler) macro, Q_PROPERTY. Together with this feature comes a new class, QVariant, which is a union for the most common Qt data types -- this enables a single mechanism to handle the property settings.

I've been told that the original intention with Q_PROPERTY was to make it easier to create bindings of Qt to scripting languages -- however, neither PyQt nor PerlQt, the currently best developed bindings, really need the feature. Indeed, it is quite possible to use functions everywhere, instead of properties.

As it is, the property mechanism is completely superfluous, creating a second way of doing the same thing, and I find the QVariant class to be a monstrosity that keeps reminding me of Visual Basic's Variant data type. I think Qt would have been cleaner and more usable without it.

New table widget is hot; others so-so

A great addition, and one that hasn't received nearly enough press, is the new table widget. This is finally an editable spreadsheet-like control. Users of Qt have been clamoring for this for years, as it is absolutely essential for all kinds of business and database applications. No longer have people to work with the difficult old QTableView, which was prone to all kinds of nasty bugs.

In the category "nice, but I don't need it" falls the QDial widget, a rounded range control like a speedometer.

The QMultiLineEdit widget is still relatively underpowered -- it doesn't support rich text and is completely line oriented, being built on QTableView. For Qt 3.0, a new and fancy QTextEdit widget is being promised, which will support rich text just like the display-only QTextView widget that is included in Qt 2.2. KDE developers have an advantage here, as they can embed the powerful KWrite text-editor widget or the KWord word processing widget in their applications through the KParts mechanism.

Interface improvements

The widget appearance has improved -- the Motif style is no longer exactly Motif, but a bit more elegant. An SGI-like style has been added, and a GTK-like style. Both reproduce the look and feel of these toolkits very well. My favourite style is still Platinum, which evokes the updated Mac look introduced with Mac OS 8. And it's as easy as ever to create new, fast widget-themes yourself.

There is now support for cute fade and scroll effects for popup and MDI windows. MDI is relatively new too -- Qt 2.1 introduced the QWorkspace class that makes it possible to write windows-style MDI applications. The trend in user-interface design is going to SDI everywhere, with one window for each document, but MDI is certainly very useful for a large class of applications, such as IDEs.

License issues

Troll Tech has drawn much criticism for their licensing approach in recent years. When they first made their library available for free software development years ago, I interpreted it as a gesture meant to support free software, not as a nasty corporate plot to get as many free demos as possible. But this was long ago -- in a time when people were actually using binary-only toolkits like XForms.

The success of the KDE desktop has changed that, and Troll Tech has finally decided to make Qt for Unix/X11 available under the GPL license. This is important, because it should help dampen a lot of unfair criticism, but in practice, little changes. The only new "right" is to distribute modified versions of Qt instead of distributing patches. Since we're talking GPL instead of LGPL, developers of closed source software still have to buy a developers license from Troll Tech, and until someone takes the GPL version of Qt and ports it independently to Windows, nothing changes for Windows developers, either.

The addition of a GPL license for Qt Embedded can become quite important, though. Even though X11 is still serviceable, it is showing its age, and some widely desired effects. such as anti-aliased font handling, are difficult to implement. Qt Embedded, which runs just as well without X11, can change that. Imagine a KDE 2 desktop that runs on Qt Embedded -- it would look very professional, won't carry any excess baggage, and would be an attractive proposition for vendors of distributions geared towards beginners.

References:

Troll Tech

KDE

PyQt

Related Articles:

Qt Designer

Qt Attracting Interest Among Application Developers

Boudewijn and Cameron Argue for Qt

Final thoughts

Qt has always been a cleanly engineered, full-featured toolkit. In the past, people have complained about its look-and-feel (although at the time when Qt offered a choice between Windows and Motif look-and-feel, no other toolkit existed that gave as much choice), but with 2.2.2, that should be over. Another serious objection to Qt -- the lack of a professional, extendable GUI designer -- has been answered too. By offering support for Unicode, localisation, and

internationalisation both on Unix and Windows, Qt provides a strong basis for modern application development. Besides the rich set of widgets, Qt now offers many utility classes with well-thought-out interfaces that are easy and safe to use.

In sum, Troll Tech have created a complete environment where developing cross-platform applications in C++ has become as easy, if not easier, as developing in Java.

Boudewijn Rempt is a senior developer with Tryllian, a mobile agent company. He has worked with PyQt since its inception.

Discuss this article in the O'Reilly Network Forum.

Return to the O'Reilly Network Hub.

Published on The O'Reilly Network (http://www.oreillynet.com/)
http://www.oreillynet.com/pub/a/python/2000/11/22/xmlrpcclient.html

XML-RPC in Python

by Dave Warner
11/22/2000

The Internet, with its simple protocol and ubiquity, has opened up huge opportunities for programs to communicate between computers, a task that always seemed complex and daunting in the past. Now there is a new dilemma. Which framework should you use for automating communication?

Related:

Web Client Programming in Python

Meerkat: The XML-RPC Interface

The two most popular frameworks, Enterprise JavaBeans and Microsoft's COM, are highly expressive and robust, but also complex. They carry an enormous amount of processing overhead. Other requirements, such as cross-platform and cross-language capability, only add to the dilemma. The Simple Object Access Protocol (SOAP) has recently come to the forefront, promising to solve many of these problems. In active development, SOAP has several partial implementations available but is still a moving target. Some implementations cannot yet talk to each other.

So what's a developer to do?

What many developers have done is to use what is in place and what works now. And what works is XML-RPC. It's easy, quick, and consistent. XML-RPC fits a "glue" role that its big brother, SOAP, cannot. If you are looking for a large-scale framework for communicating between diverse servers and enterprise functionality like automated discovery of services, then XML-RPC will not fit. XML-RPC is, however, a quick, easy-to-use, reliable means of communicating between distributed locations, without worrying about what language the "other guy" has used. XML-RPC is a hybrid solution. It's not strictly object-oriented. The "RPC" portion of the acronym stands for Remote Procedure Call. RPC is very simple -- a user sends a request over the network to a specific location and receives a reponse. If this sounds suspiciously familar, it is -- the Web works this way. The death-knell for RPC was heard in the 1990s. Its developers were unable to standardize on the structure of data. Sun did try to implement a standard in XDR. It never quite caught on. With the rise of object-oriented programming, RPC fell into disfavor. By adding XML to the RPC portion, a standard, popular structure for data has been added to the process, breathing new life into RPC. So why use it?

The major, incontestable reason for using XML-RPC is that it works, and works well. With a very minimum of effort, a developer can install and use Python xmlrpclib, communicating with servers built in Python, PHP, Java, Perl, C++, and even COM. An entire chapter devoted to XML-RPC in the best selling "Java and XML" brought great attention to XML-RPC. In the Python world, the largest use of XML-RPC is via Zope. Zope has been able to send and respond to XML-RPC since mid-1999.

Enter PythonWare's xmlrpclib

Unlike other languages in its generation, Python has always had the ability to create persistent objects. It uses increasingly complex mechanisms to do so. These mechanisms range from the standard library package, "marshall," which converts simple objects for storage in flat files, to the "Zope Object Database," which rivals commercial Object-Oriented Databases (OODB) in its robustness. To "marshall" an object is to convert it into a different format. Within the context of xmlrpclib, marshalling a Python object means to convert it into an XML-RPC call. Unmarshalling is the reverse of that step.

Written by Fredrik Lundh of PythonWare, xmlrpclib hides all of the complexity in marshalling and unmarshalling objects. It goes beyond this to also hide the details of connecting to a server, sending a request, and receiving a response.

You can download the latest version of xmlrpclib.py (and get more information about the library) at the XML-RPC for Python page. This version was last updated in June 1999, but, with a couple of minor changes (see sidebar), it works flawlessly with the latest Python and XML-SIG packages available.

XML-RPC and Meerkat

Rael Dornfest, a driving force in the adoption of XML in the PHP arena, has quite naturally constructed an XML-RPC interface to Meerkat, an open wire service that collects news information from many sources using RSS. Let's use this readily available server for our examples. First we'll import the pretty print (`pprint`) module from the Python Standard Library. Some of the output from Meerkat is quite lengthy.

```
>>> from pprint import pprint
```

After importing xmlrpclib, connecting to the server using xmlrpclib consists of one line:

```
>>> meerkatsvr =
xmlrpclib.server("http://www.oreillynet.com/meerkat/xml-rpc/server.php")
```

Here we create an instance of an XML-RPC server that understands how to connect, send, and receive XML-RPC. Through recursive "sleight-of-hand," this server can even understand nested XML-RPC requests.

Next, let's request a list of the methods that are available from the server:

```
>>> pprint(meerkatsvr.system.listMethods())
['meerkat.getChannels',
'meerkat.getCategories',
'meerkat.getChannelsByCategory',
'meerkat.getItems',
'system.listMethods',
'system.methodHelp',
'system.methodSignature']
```

You have just used XML-RPC to connect to a server across the Internet and retrieve a result set. Besides the simplicity of the approach, notice that the form of the result set returned by xmlrpclib is a native Python list.

by Dave Warner

Getting a list of channels from Meerkat

Meerkat groups RSS channels into topics, called categories. For example, channels related to XML are in category 23. If you want to get a list of channels for a specific topic, you can query Meerkat this way:

```
>>> pprint(meerkatsvr.meerkat.getChannelsByCategory(23))
[{'title': '<XML>fr', 'id': 2991},
{'title': '4xt', 'id': 2559},
{'title': 'Eclectic', 'id': 555},
{'title': 'eXploringXML', 'id': 4471},
{'title': 'eXploringXML Channel', 'id': 1105},
{'title': 'finetuning com', 'id': 107},
{'title': 'finetuning.com', 'id': 4628},
{'title': 'Free XML tools', 'id': 906},
{'title': 'JabberCentral Recent Jabber News', 'id': 4655},
{'title': 'Moreover XML and metadata news', 'id': 2243},
{'title': 'moreover... XML and metadata news', 'id': 683},
```

```
{'title': 'My Userland', 'id': 1945},
{'title': "O'Reilly Network XML FAQs", 'id': 2365},
{'title': 'oreillynet.xml', 'id': 989},
{'title': 'oreillynet.xmldev', 'id': 990},
{'title': 'SOAP Webservices Resource Center', 'id': 2022},
{'title': 'XML About com', 'id': 2435},
{'title': 'XML News from PerlXML.com', 'id': 718},
{'title': 'XML XSL Portal', 'id': 4460},
{'title': 'XML.com', 'id': 47},
{'title': 'XML.com Resource Guide', 'id': 4637},
{'title': 'xmlhack', 'id': 724},
{'title': 'xmlTree Newsletter', 'id': 413}]
```

Deeper into XML-RPC and Meerkat

In my web client programming article, I demonstrated how we could easily download a listing of all Meerkat articles related to Linux, using Python's powerful URLLIB and a very simple script. Let's revisit that task using xmlrpclib. Employing an even simpler script, we can eliminate some of the ambiguities and awkwardness of the former approach and make a more robust application.

Fire up a new interactive session in Python, import xmlrpclib, and create a server instance for interacting with the Meerkat server:

```
>>> import xmlrpclib
>>> meerkatURI = http://www.oreillynet.com/meerkat/xml-rpc/server.php)
>>> meerkatsvr = xmlrpclib.Server(meerkatURI)
```

Before we begin coding our updated solution, let's explore some features Rael Dornfest has built into the Meerkat XML-RPC server to document its capabilities. We've seen system.listMethods(). It returns an up-to-date list of the methods available on the server. You can obtain the method signature of any of these methods by calling methodSignature()

```
>>> meerkatsvr.system.methodSignature(methodname)
```

This returns a two-element array containing the return value and parameter(s) of the method named. Note that you should not include enclosing parentheses after the method name.

Dornfest also provides a means of obtaining help about any method on the server

```
>>> meerkatsvr.system.methodHelp(meerkat.getItems)
```

To see these and all the other methods available to XML-RPC clients of Meerkat, go to the Meerkat XML-RPC Interface Test page. Selecting introspection from the drop-down list will give you all methods available, with their documentation. The URI, as its name suggests, also tests the Meerkat XML-RPC Server, returning both the method call and the server response. Nice.

Obtaining a list of Linux articles from Meerkat

As you may recall, the objective of the program in my web client programming article was to obtain links to Linux articles from Meerkat over the past hour. After building the framework to access the server through URLLIB, we constructed the URI:

```
http://www.oreillynet.com/meerkat/?p=5&t=1HOUR&_fl=minimal&_de=0&_ca=0&_ch=0&_da=0.
```

Imagine having to maintain the above line in an application! As a refresher, the URI tells Meerkat to give us all stories in the Linux category that have been posted in the last hour, using the minimal flavor. We want to further restrict the information coming back by eliminating the description, category, channel, and date fields. Let's replicate this functionality using xmlrpclib.

Declare a parameter list by constructing a Python dictionary (the same as a structure in XML-RPC):

```
>>> params = {
... 'category' : 7,
... 'time_period' : '1HOUR',
... 'descriptions' : 0,
... 'categories' : 0,
... 'channels' : 0,
... 'dates' : 0
    }
```

The first two parameters define the scope of the information we desire. The last four act as switches, turning off the various components returned. Next, we'll call a method on the server to obtain the desired information

```
>>> results = meerkatsvr.meerkat.getItems(params)
```

This returns a list of dictionaries containing both the title and an HTML link for all Linux articles that have appeared on Meerkat over the last hour. If your request returns an empty list, increase the time period (e.g. 2HOUR or 1DAY).

Now, let's loop through this list and construct full-fledged HTML links to the articles:

```
>>> for d in results:
...     print <A HREF=%s>%s</A> % (d[link],d[title])
```

What we have gained

This approach has given us several benefits. We have enhanced readability. We have less code. Presentation has been separated from data, allowing you to choose alternatives easily. Best of all, this approach has a distinctly Pythonic feel to it. Thanks to the functionality built into XML-RPC and xmlrpclib, the response from Meerkat is now Python data, with all the attendant flexibility: Pickle it, store it in a database, show it in a wxPython or Tkinter widget. Or even send it to another client via XML-RPC!

In the next article, I will dig deeper into the internals of xmlrpclib to show how Python objects are converted into XML-RPC requests and how return values are unmarshalled. We will also build an XML-RPC server that includes a testing framework.

Dave Warner is a senior programmer and DBA at Federal Data Corporation. He specializes in accessing relational databases with languages that begin with the letter P: Python, Perl and PowerBuilder.

Related:

Web Client Programming in Python

Meerkat: The XML-RPC Interface

Discuss this article in the O'Reilly Network Python Forum.

Return to the Python DevCenter.

Published on The O'Reilly Network (http://www.oreillynet.com/)
http://www.oreillynet.com/pub/a/wireless/2000/11/22/wireless_plamvx.html

by Derrick Story
11/22/2000

Free Starbucks coffee is served every morning in the lobby of San Francisco's Hotel Monaco. Of course, you have to stay there to drink it. Normally, I'm accompanied by my Handspring Visor as I sip from my cup of house blend. But today is different -- the Visor is back in the room. My drinking buddy this morning is a Palm Vx Limited Edition with a Novatel Minstrel V wireless modem.

Thanks to an innovative promotion at CMP's Web2000, I and scores of other conference attendees were able to test this state-of-the-art wireless package. And during the 24 hours that I put this setup through its paces, I evolved from basic consumer thinking (like using the wireless connection to find the next Starbuck's location) to contemplating how this tool could be used in business to increase productivity and efficiency.

But first, I want to show you the gizmo itself ...

Related:

Wireless Palms: What Are the Options?

Compaq's iPAQ PocketPC: The Multi-Purpose Multimedia Handheld

Wireless News from Meerkat

Previous Features

More from the Wireless DevCenter

The setup

The Palm Vx is amazingly svelte considering that it packs 8 MB of RAM. It's a good choice for a wireless package because the Novatel modem adds considerable size and weight to the package. Compared to my Handspring Visor, the Palm/Minstrel unit was 1/8" longer and 3/16" deeper. And it did feel heavier than my normal PDA companion.

However, if you wanted to leave the modem behind and travel light, it simply slips off and you have the light Palm Vx which rests comfortably in a shirt pocket without even causing it to sag.

The Minstrel V features cellular digital packet data (CDPD) technology, which provides real-time data transmission at speeds up to 19.2 kbps over a secure open standard network. I was provided with an IP address that allowed me to connect to a Verizon Wireless network. The modem is always on standby, and if you need to retrieve information from the Internet, it automatically connects. The only way to

turn off the modem is by launching the Minstrel software application and selecting shutdown.

Since both the Palm V and the modem are rechargeable, worrying about the unit remaining on isn't really an issue. Both units had enough juice to get me through a 14-hour day. And I simply plugged them into their respective charging units before dozing off for the evening.

For browsing, the Palm Vx included AvantGo with 14 preloaded channels ranging from ABC News to a five-language translator. This was a different face to AvantGo that I hadn't experienced before. Normally, I download the weather, daily news, and tech updates as I dash out the door and read them during the day as time allows. (This is one of my favorite PDA activities.) But now, with the wireless setup, I didn't have to worry about preloading. Whenever I wanted the day's top stories, I'd open AvantGo on the Palm and it would go grab them in real time.

Palm Vx with Novatel Minstrel V modem.

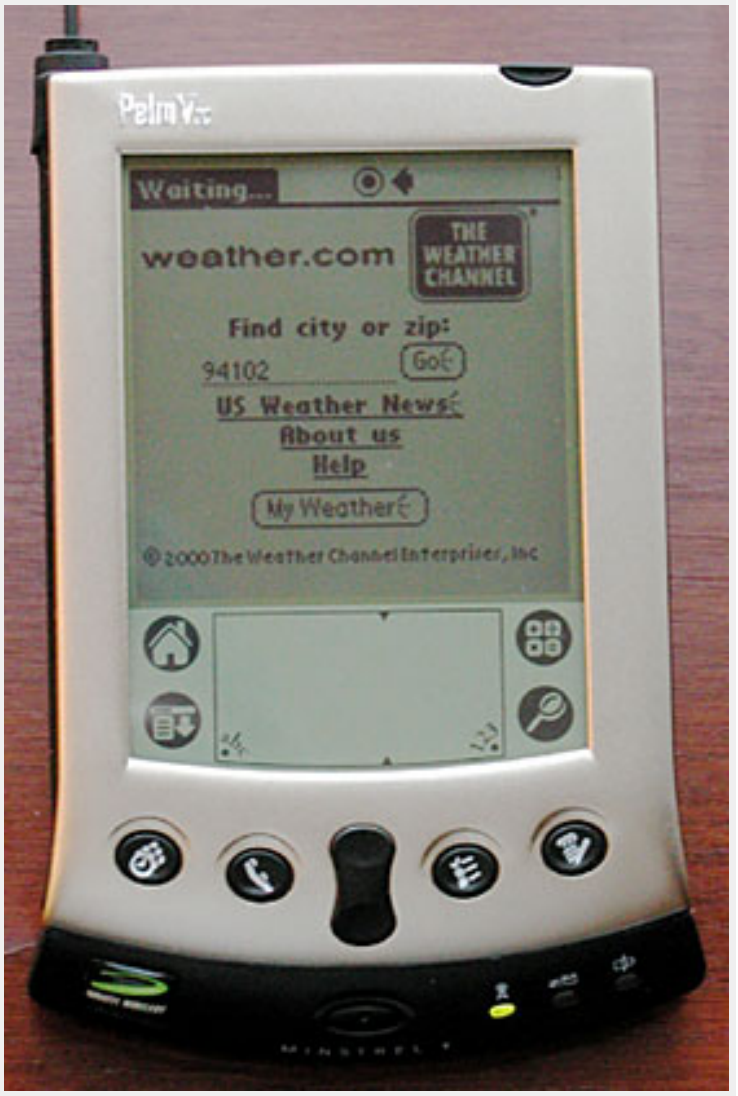
Is this real-time benefit mission-critical to my life? Hard to say in just the 24 hours I used the device. On one hand, it was handy getting the weather report for "right now" before I dashed outside without a coat. On the other hand, the process of logging on the Verizon server and retrieving the information took considerably longer than checking my preloaded AvantGo channels. More on this later.

For e-mail, I used the excellent MultiMail 3.1, which is better designed and more reliable than most desktop clients. Since I already use this application on my Visor with a ThinModem module, I was able to quickly configure it and download my mail. By the way, one of the great functions of this app is the option to download "headers only." This prevents your PDA from being lobotomized by huge, mind-numbing e-mail files. I simply scan the headers, grab the messages of the ones I want, and delete the rest.

A copy of the Blogger Wireless Edition was also pre-installed. This innovative PDA app allows you to post and read blogs on your Palm. With the wireless edition, you could literally blog while commuting on public transportation or waiting for your lunch to arrive in a crowded restaurant. The wireless edition offers a subset of functionality found at www.blogger.com: account creation, blog creation, posting, editing, publishing, and a directory of blogs suitable for viewing on a PDA. For content providers on the go, this functionality alone could justify the investment for the wireless package.

A typical consumer scenario -- thinking inside of the box

Back to my cup of coffee at the Hotel Monaco. After a few sips, I set down the cup and pulled up the antenna on the Minstrel modem. First stop, the weather channel. I typed in the 94102 zip code and was greeting with a partly cloudy, calm wind, 55°F report. Today's high was predicted for 72° -- perfect!



Checking the weather in real time.

Next I checked the traffic and was happy to read that there were no accidents between the hotel and the Moscone Conference Center. However, I was sad to read that a Muni bus had crashed into a building at the corner of Battery and Broadway. Fortunately, no major injuries were reported, and the mess was expected to be cleared by 9:28 a.m.

ABC news reported that a car bomb had exploded in Jerusalem and that the dot.coms are dead. (I'm still weighing the merits of having access to real-time news ...)

I then located an ATM machine just three blocks away, right next to another Starbucks. Lastly, I checked my e-mail and logged off.

At this point, none of this functionality is really that different than what I could do with my considerably less expensive Visor setup. With it, I would download the same AvantGo information on my way out the door and read it as I had time over the course of the morning. "I need to expand my thinking here," I thought. "There's got to be more to wireless than traffic reports and depressing news."

Custom business applications -- thinking outside of the box

Later that day, I interviewed Bridget Hart, the Web2000 show director and project leader for the "try and buy" wireless promotion. She told me that this technology isn't just for finding the nearest Starbucks. (At this point I quickly scrolled the Starbucks locator off my screen and nodded in agreement.)

"I'm excited about this promotion, the largest of its kind ever, because it allows us to start thinking about the possibilities when using wireless with handheld devices," she said. "For example, how could we use these new tools in the corporate setting to improve efficiency? Wireless LANs are great, but they have a limited range. What if we could upload information in real time that our coworkers could

be using right away instead of at the end of the day? I think the possibilities are limitless."

I then met with Handango Product Manager Brad Ellison to discuss how his company is providing business solutions to its corporate customers. Handango is mainly in the PDA software business with a library of 8,500 titles from 3,300 software developers. But they also specialize in pulling together complete corporate solutions including hardware and connectivity. They coordinated, for example, the Web2000 try-and-buy promotion.

According to Mr. Ellison, businesses are starting to explore these tools for use in the corporate environment. He pointed me to a partnership Handango entered into earlier this year with TRGpro devices where the two companies are going to work together to provide complete business solutions.

The Web2000 custom business application

The Web2000 group also worked with Centura Software to create a customized application dubbed "The Web2000 Brain" that provided conference attendees with scheduling, messaging, and networking capabilities. With it, I could check class times, read show dailies, exchange opinions with other attendees, and locate exhibitors. I loved not having to lug the paper version of the conference program around just to look something up.

Even though I enjoyed the functionality of the Web2000 Brain, it wasn't until that evening that it occurred to me that this was a working example of the types of custom apps that any business could create and implement. Companies such as Centura specialize in creating the middleware that is often the "missing piece of the puzzle" for businesses.

Once I began to appreciate that the Web2000 Brain was actually a custom application designed for the "business" (the Web2000 Conference), I began to think about how I could apply this technology in my own business life.

The cost of going wireless

All of these possibilities come at a price, however. The folks at Handango put together a convenient package for Web2000 attendees that read like this:

- Palm Vx Limited Edition -- $399
- Novatel Minstrel V Modem -- $345
- Monthly wireless service -- $39.95 a month
- Software package -- $125

There are more and more options appearing all of the time, but clearly this isn't an "impulse buy" type of purchase.

Bringing it home

It's been a couple weeks since Web2000, and I've happily returned to using my Visor with AvantGo and a ThinModem module. I haven't really yearned much for wireless freedom -- even after having tasted the fruit.

But I have continued thinking about the possibilities. How much more efficiency could I bring to my business world with a wireless PDA? Would I (someday soon) be sitting in the backseat during a long commute posting a blog -- or catching up on my e-mail? As technology continues to improve and as prices come down, wireless Internet may become a normal part of my life. For now, however, I think I'll just enjoy the ride to work and chat with my driving companions.

Derrick Story is the managing editor of the O'Reilly Network.

Resources

Handango is one-stop shopping for handheld computing.

Centura creates the middleware that is often the missing link for business applications.

Related:

Wireless Palms: What Are the Options?

Compaq's iPAQ PocketPC: The Multi-Purpose Multimedia Handheld

Wireless News from Meerkat

Discuss this article in the O'Reilly Network Wireless Forum.

Return to the Wireless DevCenter.

Published on The O'Reilly Network (http://www.oreillynet.com/)
http://www.oreillynet.com/pub/a/bsd/2000/11/22/FreeBSD_Basics.html

FreeBSD basics by Dru Lavigne

Discovering System Processes Part II

11/22/2000

In last week's article, we took a look at what processes are and how to view the running processes on your FreeBSD system. In this week's article, I'd like to look at how processes talk to each other, how you can talk to your processes, and why you might want to do so.

When processes talk to each other, it is called interprocess communication. Processes aren't allowed to say just anything to each other; instead, your FreeBSD system comes with 31 possible predefined messages. These messages are called signals; you can view these signals with this command:

```
kill -l
```

```
HUP INT QUIT ILL TRAP ABRT EMT FPE KILL BUS SEGV SYS PIPE ALRM TERM URG
STOP TSTP CONT CHLD TTIN TTOU IO XCPU XFSZ VTALRM PROF WINCH INFO USR1
USR2
```

Each signal has a numerical equivalent, and the signals are listed in numerical order; therefore, `HUP`=1, `INT`=2, etc. Processes are allowed to send these signals to each other; users are also allowed to send these signals to processes.

To find out what each of these signals does, read sigaction(2) by typing `man 2 sigaction` or signal(3) by typing `man 3 signal`. The following chart summarizes each of the 31 signals, their resultant actions, and a brief description of each. Although users can send any of these signals, I've included an asterisk next to the signals most commonly used by users.

Signal Name	#	Default Action	Description
*HUP	1	terminate process	terminal line hangup
*INT	2	terminate process	interrupt program
*QUIT	3	create core image	quit program
ILL	4	create core image	illegal instruction
TRAP	5	create core image	trace trap
ABRT	6	create core image	abort(3) call

EMT	7	create core image	emulate instruction executed
FPE	8	create core image	floating-point exception
*KILL	9	terminate process	kill program
BUS	10	create core image	bus error
SEGV	11	create core image	segmentation violation
SYS	12	create core image	non-existent system call
PIPE	13	terminate process	write on a pipe with no reader
ALRM	14	terminate process	real-time timer expired
*TERM	15	terminate process	software termination signal
URG	16	discard signal	urgent condition
*STOP	17	stop process	stop (cannot be ignored)
*TSTP	18	stop process	stop keyboard signal
CONT	19	discard signal	continue after stop
CHLD	20	discard signal	child status has changed
TTIN	21	stop process	background read attempted
TTOU	22	stop process	background write attempted
IO	23	discard signal	I/O is possible
XCPU	24	terminate process	CPU time limit exceeded
XFSZ	25	terminate process	file size limit exceeded
VTALRM	26	terminate process	virtual time alarm
PROF	27	terminate process	profiling timer alarm
WINCH	28	discard signal	Window size change
INFO	29	discard signal	status request from keyboard
USR1	30	terminate process	User defined signal 1
USR2	31	terminate process	User defined signal 2

Some signals are used so often by users that they have been mapped to control characters. To view your control character mappings, look at the last four lines of the output of:

```
stty -e
discard dsusp   eof     eol     eol2    erase   intr    kill    lnext
^O      ^Y      ^D      <undef> <undef> ^H      ^C      ^U      ^V
min     quit    reprint start   status  stop    susp    time    werase
1       ^\      ^R      ^Q      ^T      ^S      ^Z      0       ^W
```

The ^ symbol means to press your CTRL key at the same time as the letter that follows it. Note that three signals have been mapped as control characters:

- ^C has been mapped to the INT signal, or signal 2
- ^\ has been mapped to the QUIT signal, or signal 3
- ^Z has been mapped to the TSTP signal, or signal 18 (even though it's called susp here)

Don't confuse the word kill in this output with the KILL signal (signal 9). ^U will erase an entire line, not send a signal 9. To prove this to yourself, type a long string of words at your prompt, then do a ^U.

So, how do you send signals that haven't been mapped to control characters? Use the kill command.

```
whatis kill
kill(1)   - terminate or signal a process
kill(2)   - send signal to a process
```

There's a couple of different ways to use kill; if you simply type:

```
kill PID
```

the default TERM signal will be sent to the process with the PID that you indicated. If you wish to change the type of signal to send, you can specify the signal either by its name or by its number like so:

```
kill -signal_name PID
```

or

```
kill -signal_number PID
```

So:

```
kill PID
kill -TERM PID
```

and

```
kill -15 PID
```

are equivalent commands. Remember that Unix is case-sensitive; if you type:

```
kill -term PID
```

you'll receive the following error message:

```
term: Unknown signal; kill -l lists all signals.
```

So, now that we've seen the 31 possible messages and how to send them, let's look at examples of reasons you would want to send a signal to a process. Often when you're working your way through the FreeBSD handbook or a tutorial, you'll learn how to make changes to a configuration file and will then be prompted to send a HUP signal. Most processes only read their configuration file when they first start up; the HUP signal tells a process to stop running; when that process restarts, it will reread its configuration file and your changes to that file will take effect. Also, every time you log out of a terminal, a HUP signal is sent to all processes running on that terminal; this means that all processes that were running on that terminal will be stopped.

by Dru Lavigne

Sometimes you may start a process and wish to stop it before it is finished. For example, in a spurt of inspiration you might decide that you want to see the name of every file on your FreeBSD system, so you

type this at your terminal:

```
find / -print |more
```

However, you soon grow tired of pressing the spacebar and decide that you really didn't want to see all of your files at this time. In other words, you want to send an interrupt signal. One way to do this is:

```
^C
```

You'll know that your INT signal worked as you'll get your prompt back.

Retry the same `find` command, but this time send a signal 3 like so:

```
^\
```

Just before you get your prompt back, you'll see the following message:

```
Quit  (core dumped)
```

If you use ALT F1 to return to the console, you'll see a message similar to this:

```
Nov 19 13:50:09 genisis /kernel: pid 806 (find), uid 1001: exited on
signal 3
Nov 19 13:50:09 genisis /kernel: pid 807 (more), uid 1001: exited on
signal 3 (core dumped)
```

And if you do a directory listing at your original terminal, you should see a file called `more.core`. Normally, you won't be sending a signal 3 to a process unless you're a programmer and know how to use the kernel debugger. I included the example to show the difference between a signal 2 and a signal 3; you can safely delete that `*.core` file.

Interprocess communication isn't much different than any other type of communication. You or another process can send a signal requesting a desired result, but it is up to the process receiving the signal to decide what it wants to do with that signal. Remember that processes are simply running programs; most programs use something called a "signal handler" to decide how and when to respond to signals. Usually if you send some type of termination signal, the signal handler will try to gracefully close all the files that process has opened to prevent data loss before the process itself closes. Sometimes, the signal handler will decide to just ignore your signal and will refuse to terminate the process.

However, some signals can't be ignored; for example, signal 9 and signal 17. Let's say that you wish to stop a process you've started, so you used `grep` to find the PID of the process, used `ps` to send a `TERM` signal, then repeated your `grep` to ensure it worked like so:

```
ps | grep processname

kill PID

ps | grep processname
```

However, the second `grep` still shows that PID, meaning your `TERM` signal was ignored by that process. Either one of these commands should fix it:

```
kill -9 PID
```

or

```
kill -KILL PID
```

If you now repeat your `grep` command, you should just have your prompt echoed back at you, meaning that PID was indeed terminated.

You may ask, "Why not just always send a signal 9 if it can't be ignored?" Signal 9 does indeed "kill" a process, but it doesn't give it time to gracefully save all of its work first, meaning that you may lose some data. It's better to try sending another type of terminating signal first, and save signal 9 for those processes that stubbornly refuse to terminate. Also, remember that as a regular user you will only be able to send signals to processes that are owned by you. The superuser can send a signal to any process.

There may be times when you wish to terminate all the processes you own; this has different ramifications depending on whether you are a regular user or the superuser.

Let's demonstrate as a regular user. Log in to four different terminals and do a `ps` command:

```
ps
  PID  TT  STAT      TIME COMMAND
  316  v0  Ss     0:00.39 -csh (csh)
  957  v0  R+     0:00.00 ps
  317  v1  Is+    0:00.20 -csh (csh)
  915  v2  Is     0:00.12 -csh (csh)
  941  v2  I+     0:00.09 lynx
  942  v2  Z+     0:00.00  (lynx)
  913  v3  Is     0:00.12 -csh (csh)
  946  v3  I+     0:00.01 /bin/sh /usr/X11R6/bin/startx
  951  v3  I+     0:00.04 xinit /home/genisis/.xinitrc --
  955  v3  S      0:03.00 xfce
```

In this example, I've logged into terminals 0-3. I ran the `ps` command from the console, logged into the first terminal, started `lynx` on the second terminal, and started an XWindows session from terminal three, which resulted in a total of 10 processes owned by myself. If I use a PID of "-1" when I invoke the `kill` command, I will broadcast the signal I specify to all of my processes. So, let's send a `TERM` signal like so:

```
kill -1
```

Then check our results with the `ps` command:

```
ps
  PID  TT  STAT      TIME COMMAND
  316  v0  Ss     0:00.41 -csh (csh)
  969  v0  R+     0:00.00 ps
  317  v1  Ss+    0:00.21 -csh (csh)
  915  v2  Is+    0:00.12 -csh (csh)
  913  v3  Is+    0:00.12 -csh (csh)
```

Looks like we terminated six of the original PIDs, but four processes ignored our `TERM` signal. Let's be a bit more aggressive:

```
kill -KILL -1
  PID  TT  STAT      TIME COMMAND
```

```
  317  v1  Ss     0:00.22 -csh (csh)
  995  v1  R+     0:00.00 ps
```

If you scroll through your original four terminals, you'll see the login prompt at three of them. This last command killed all processes except the process you executed the `kill` command from, that is, all processes except the c shell you ran the `kill` command in.

You'll note that if you make a typo and type:

```
kill 1
```

instead of:

```
kill -1
```

you'll receive the following error message:

```
1: Operation not permitted
```

`-1` is the special PID that represents all of your processes; `1` is the PID of the process named `init`. Only the superuser can kill the `init` process. Also, the superuser should only kill `init` if the superuser knows what he is doing.

Now let's see what happens if we repeat this exercise as the superuser. First, I'll run the `ps` command on my test computer that is running all kinds of neat stuff: Apache, MySQL, Squid, NFS, etc.

```
ps -acux
```

```
USER      PID %CPU %MEM   VSZ  RSS  TT  STAT STARTED      TIME COMMAND
genisis  1050  0.0  0.2   428  244  v0  R+    4:08PM   0:00.00 ps
root        1  0.0  0.2   532  304  ??  ILs   5:10AM   0:00.04 init
root        2  0.0  0.0     0    0  ??  DL    5:10AM   0:00.03 pagedaemon
root        3  0.0  0.0     0    0  ??  DL    5:10AM   0:00.00 vmdaemon
root        4  0.0  0.0     0    0  ??  DL    5:10AM   0:00.04 bufdaemon
root        5  0.0  0.0     0    0  ??  DL    5:10AM   0:02.62 syncer
root       27  0.0  2.0 70780 2540  ??  ILs   5:10AM   0:00.08 mount_mfs
root       30  0.0  0.1   208   92  ??  Is    5:10AM   0:00.00 adjkerntz
root      110  0.0  0.3   536  368  ??  Ss   10:10AM   0:00.22 dhclient
root      163  0.0  0.5   904  608  ??  Ss   10:10AM   0:00.19 syslogd
daemon    166  0.0  0.4   916  556  ??  Is   10:10AM   0:00.01 portmap
root      171  0.0  0.3   504  320  ??  Is   10:10AM   0:00.00 mountd
root      173  0.0  0.1   360  172  ??  Is   10:10AM   0:00.01 nfsd
root      175  0.0  0.1   352  164  ??  I    10:10AM   0:00.00 nfsd
root      176  0.0  0.1   352  164  ??  I    10:10AM   0:00.00 nfsd
root      177  0.0  0.1   352  164  ??  I    10:10AM   0:00.00 nfsd
root      178  0.0  0.1   352  164  ??  I    10:10AM   0:00.00 nfsd
root      181  0.0  0.5 263052  576  ??  Is   10:10AM   0:00.00 rpc.statd
root      197  0.0  0.6  1028  764  ??  Is   10:10AM   0:00.02 inetd
root      199  0.0  0.6   956  700  ??  Ss   10:10AM   0:00.19 cron
root      202  0.0  1.0  1424 1216  ??  Is   10:10AM   0:00.20 sendmail
root      227  0.0  0.4   876  488  ??  Is   10:10AM   0:00.00 moused
root      261  0.0  1.4  2068 1704  ??  Ss   10:10AM   0:00.98 httpd
root      275  0.0  0.4   620  448 con- I+   10:10AM   0:00.02 sh
```

```
root      293  0.0  0.4   624  452 con- I+   10:10AM   0:00.01 sh
mysql     303  0.0  1.4 10896 1796 con- S+   10:10AM   0:00.43 mysqld
nobody    305  0.0  4.7  6580 5928 con- S+   10:10AM   0:05.42 squid
nobody    308  0.0  1.4  2092 1704  ??  I    10:10AM   0:00.00 httpd
nobody    309  0.0  1.4  2092 1704  ??  I    10:10AM   0:00.00 httpd
nobody    310  0.0  1.4  2092 1704  ??  I    10:10AM   0:00.00 httpd
nobody    311  0.0  1.4  2092 1704  ??  I    10:10AM   0:00.00 httpd
nobody    312  0.0  1.4  2092 1704  ??  I    10:10AM   0:00.00 httpd
genisis   317  0.0  0.8  1336  960  v1  Is+  10:10AM   0:00.24 csh
root      320  0.0  0.5   920  628  v4  Is+  10:10AM   0:00.02 getty
root      321  0.0  0.5   920  628  v5  Is+  10:10AM   0:00.01 getty
root      322  0.0  0.5   920  628  v6  Is+  10:10AM   0:00.01 getty
root      323  0.0  0.5   920  628  v7  Is+  10:10AM   0:00.01 getty
nobody    324  0.0  0.3   832  348  ??  Is   10:10AM   0:00.01 unlinkd
root      992  0.0  0.5   920  628  v2  Is+   3:46PM   0:00.01 getty
root      993  0.0  0.5   920  628  v3  Is+   3:46PM   0:00.01 getty
genisis   994  0.0  0.8  1336  956  v0  Ss    3:46PM   0:00.14 csh
root        0  0.0  0.0     0    0  ??  DLs   5:10AM   0:00.02 swapper
```

Then I'll send the `KILL` signal to the special PID `-1` as the superuser:

```
su
Password:
kill -9 -1
```

That command was a little scarier as it even kicked me out of the c shell I executed the `kill` command from. Once I logged back in, I assessed the damage like so:

```
ps -acux
USER      PID %CPU %MEM   VSZ  RSS  TT  STAT STARTED      TIME COMMAND
genisis  1070  0.0  0.2   396  244  v0  R+    4:11PM   0:00.00 ps
root        1  0.0  0.2   532  304  ??  ILs   5:10AM   0:00.05 init
root        2  0.0  0.0     0    0  ??  DL    5:10AM   0:00.03 pagedaemon
root        3  0.0  0.0     0    0  ??  DL    5:10AM   0:00.00 vmdaemon
root        4  0.0  0.0     0    0  ??  DL    5:10AM   0:00.05 bufdaemon
root        5  0.0  0.0     0    0  ??  DL    5:10AM   0:02.65 syncer
root     1059  0.0  0.5   920  628  v3  Is+   4:10PM   0:00.01 getty
root     1060  0.0  0.5   920  628  v2  Is+   4:10PM   0:00.01 getty
root     1061  0.0  0.5   920  628  v7  Is+   4:10PM   0:00.01 getty
root     1062  0.0  0.5   920  628  v6  Is+   4:10PM   0:00.01 getty
root     1063  0.0  0.5   920  628  v5  Is+   4:10PM   0:00.01 getty
genisis  1064  0.0  0.8  1336  956  v0  Ss    4:10PM   0:00.12 csh
root     1065  0.0  0.5   920  628  v4  Is+   4:10PM   0:00.01 getty
root     1066  0.0  0.5   920  628  v1  Is+   4:10PM   0:00.01 getty
root        0  0.0  0.0     0    0  ??  DLs   5:10AM   0:00.02 swapper
```

When the superuser sends a signal to `-1`, it is sent to every process except the system processes. If that signal happened to be the `KILL` signal, you would be hearing complaints from users who happened to have a

file open at the time and lost their data.

This is one of the reasons only the superuser is allowed to run the `reboot` and `halt` commands. When one of these commands is issued, a `TERM` signal is sent to PID `-1` to give all processes a chance to save their data; this is followed by a `KILL` signal to ensure that any remaining processes are terminated.

In next week's article, I'd like to continue a bit more on this theme and take a closer look at `init` and `getty`.

Dru Lavigne is a networking instructor at a private technical college in Kingston, ON, and is notorious for seeing how many operating systems she can convince to multi-boot on her test machine.

---

*Read more from **FreeBSD Basics**.*

Discuss this article in the Operating Systems Forum.

Return to the BSD DevCenter.

Published on The O'Reilly Network (http://www.oreillynet.com/)
http://www.oreillynet.com/pub/a/p2p/2000/11/21/freenetcontent.html

by Jon Orwant
11/21/2000

I just spent some time discovering what's on Freenet, the peer-to-peer system that makes it possible to publish data anonymously and in such a way that it effectively cannot be removed. The software has a solid architecture, but feels like little more than the foundation of a castle. You can see what it might be someday, but at the moment it's not very useful: the installation process is clunky, and there aren't any polished clients. It's hard to find out what other nodes are out there; when you've found a node it's hard to find out what content you can retrieve from it; once you've found content you have no way to discern its quality.

Freenet has physics lectures by the late great Richard Feynman, the confessions of Jean-Jacques Rousseau, and the text of NAFTA. It also has an entire Sinead O'Connor album, the text of one O'Reilly book, kiddie porn, and pictures of naked people playing darts.

It would be unfair to infer the tastes of the average Freenet user from what's been uploaded (and more to the point, from the keys of the uploaded files -- there's no reason why something called constitution.txt can't actually be an MPEG movie of exotic clown dancing, sausages being made, or Florida polling booth webcam footage). But if we were to indulge ourselves and construct a demographic of the average Freenet user from Freenet content, he'd be a crypto-anarchist Perl hacker with a taste for the classics of literature, political screeds, 1980s pop music, Adobe software, and lots of porn.

Related Articles:

Free Radical: Ian Clark has Big Plans for the Internet

How Ray Ozzie Got His Groove Back

A Directory of Peer-to-Peer Projects

Open Source Roundtable: Free Riding on Gnutella

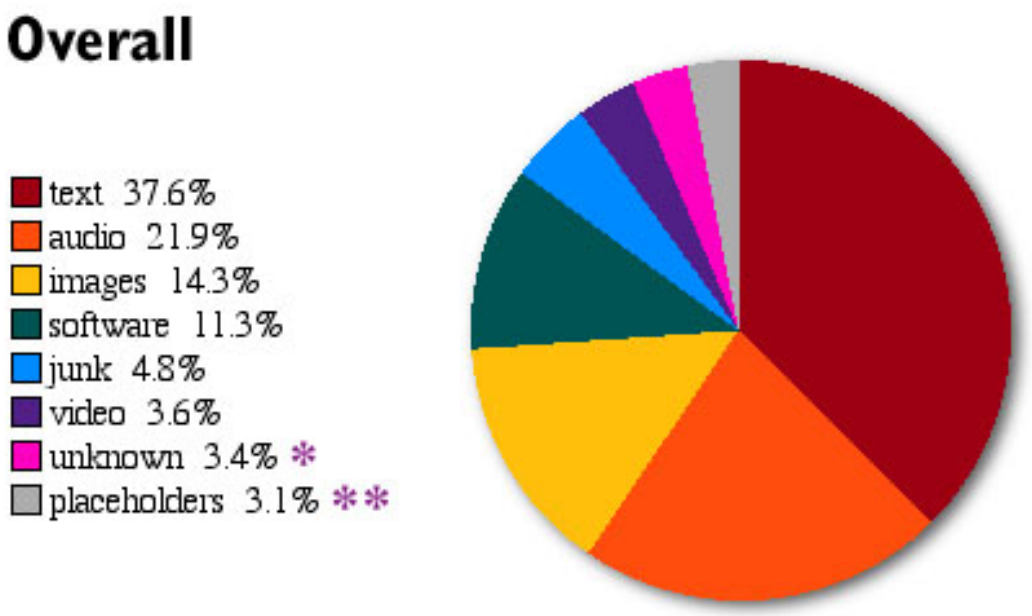
O'Reilly's Peer-to-Peer Conference

How the Peer-to-Peer Working Group Ought to Be Organized

More from the P2P DevCenter

Previous Features

I tabulated all the items I could find on Freenet (1075 items) and categorized them based on their key name.



* but seemingly meaningful
** indexes of other Freenet content

These stats tend to overreport more storage intensive media: for instance, one Ruth Rendell audio book (which was counted as audio, not as text) was split into four uploads, and so is represented by four distinct keys.

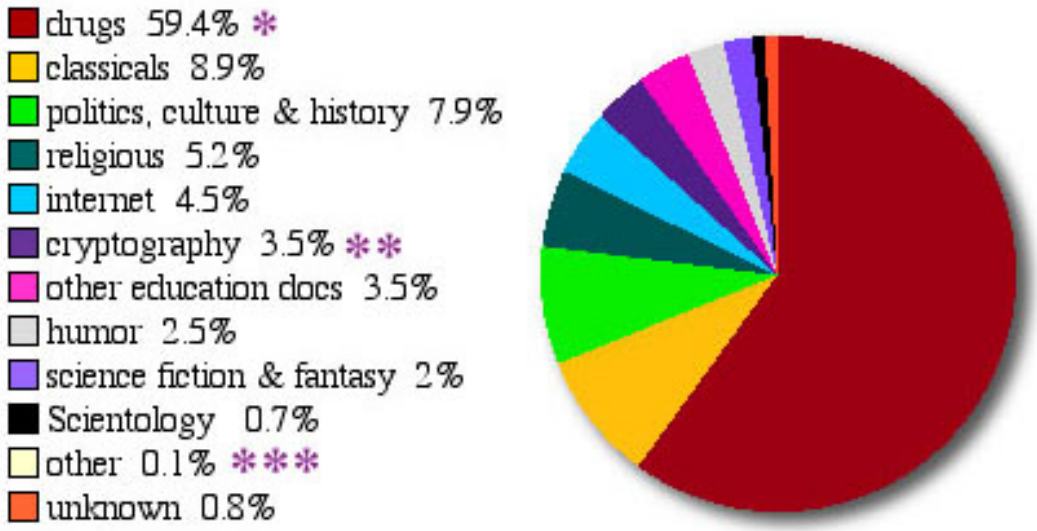
Also, several dozen items were uploaded under separate names but almost certainly contain identical content, such as `docs/books/english/fiction/george_orwell/1984.html` and `docs/books/english/fiction/george_orwell/1984.html.zip`.

Further observations within each medium:

Text

19.1% were seemingly books and 80.9% were "mere" documents, although it was often hard to draw a fine line between the two.

Breaking down the text:

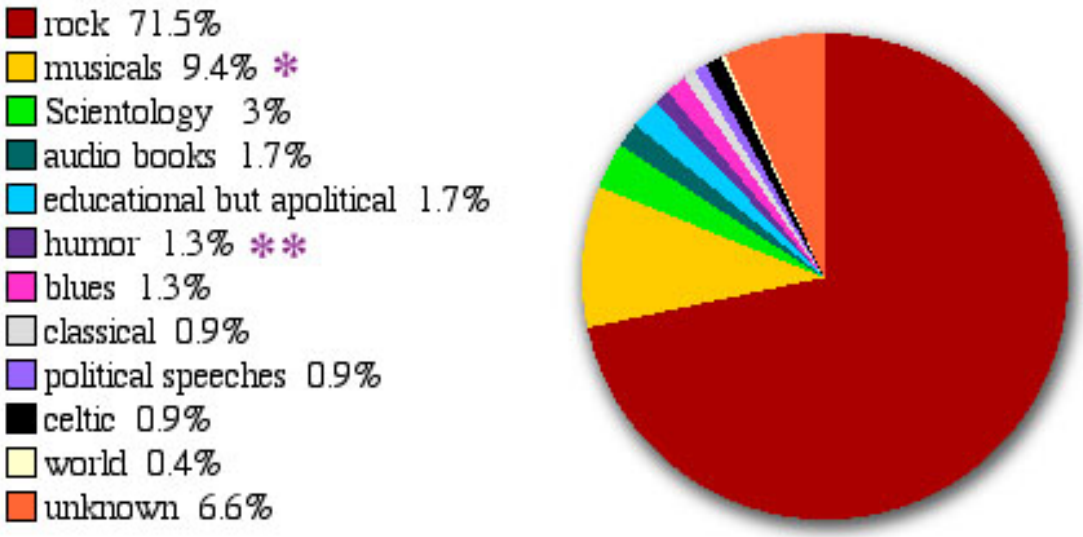


* most of them contraband in the U.S.
** including Freenet docs
*** copyright-infringing books

There was one O'Reilly book included: the PDF of *Open Sources*. That was the only computer book, unless you count the Jargon file. A Harry Potter novel was there as well, as was George Orwell's *1984*.

There was also information about the Florida ballot recount currently underway.

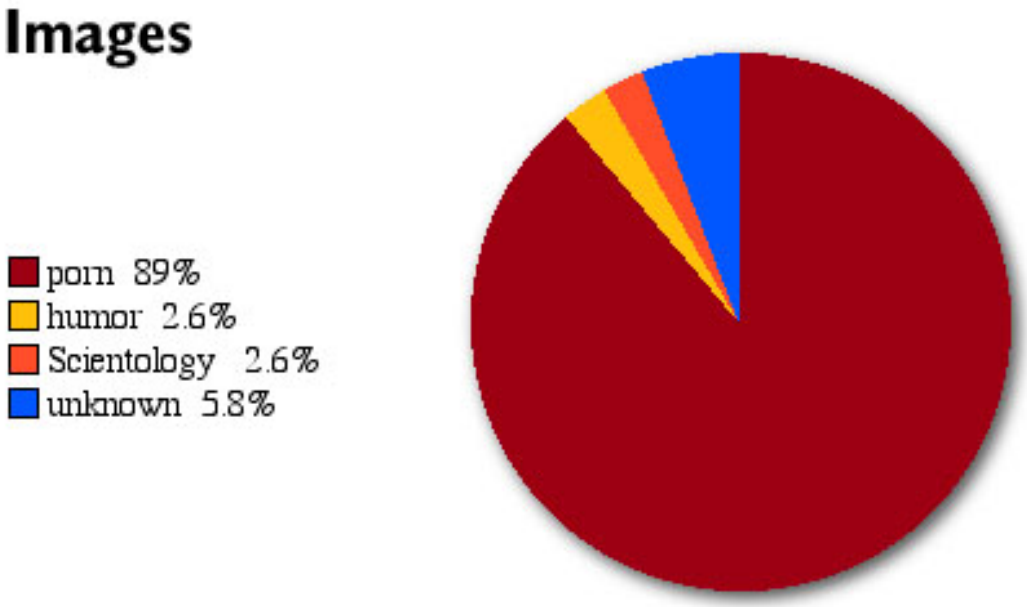
Audio



* all of them Jesus Christ Superstar
** Tom Lehrer, Weird Al Yankovic, and Monty Python

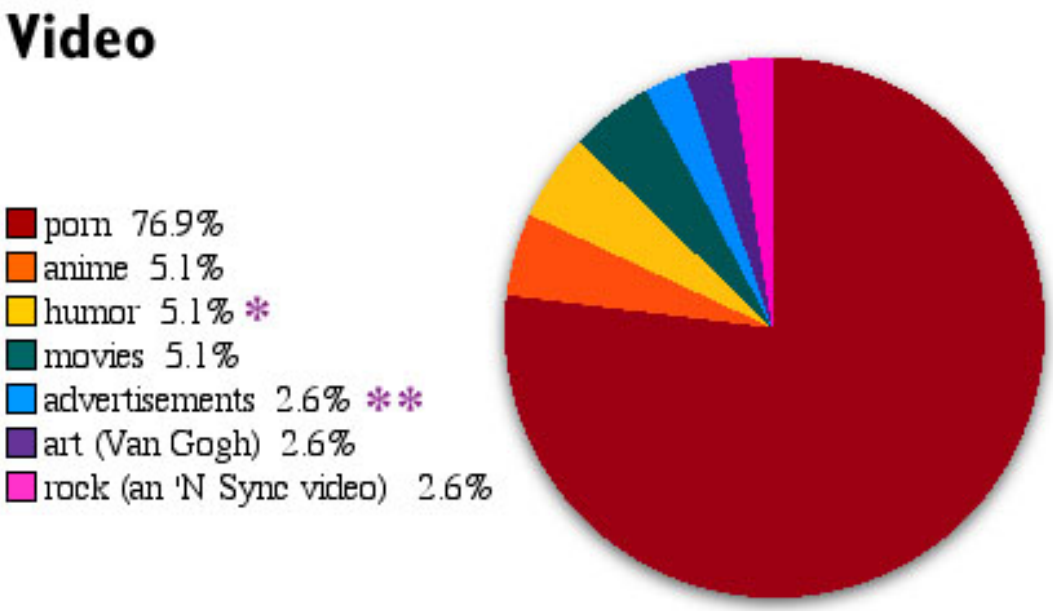
There were entire albums by some bands, including Genesis, Eurythmics, Alphaville, Stereolab, Information Society, and Sinead O'Connor.

Note that while Freenet music is definitely overwhelmingly rock, these by-key statistics will tend to over-report rock compared to, say, classical music, since the average rock album has more tracks than the average classical album.



The porn images include four images whose names imply that they are kiddie porn.

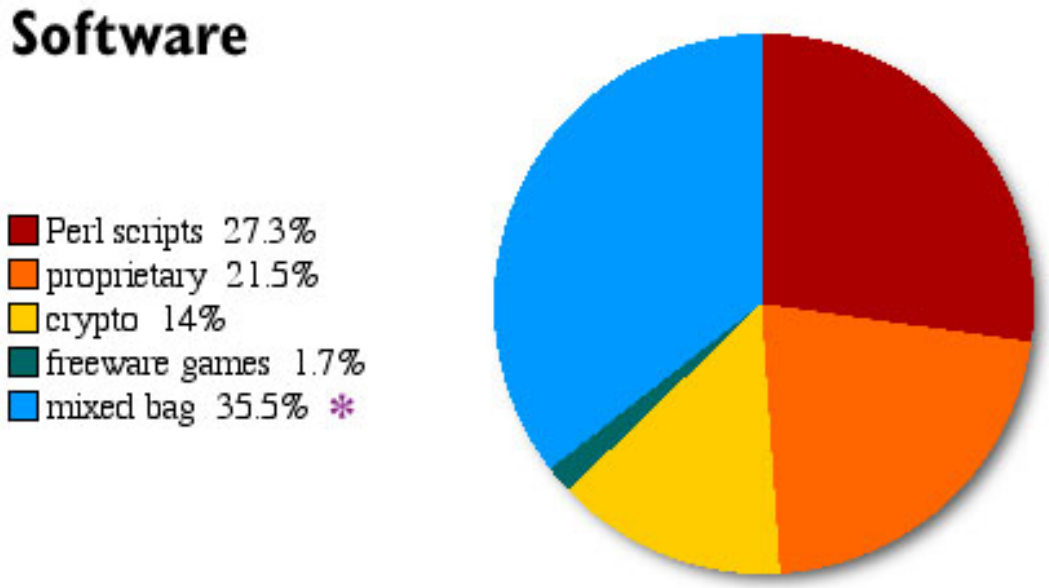
(And before you start snickering, let me take this opportunity to remind you that I'm categorizing these based only on key name. I haven't seen any of the images.)



❊ Dilbert shockwaves

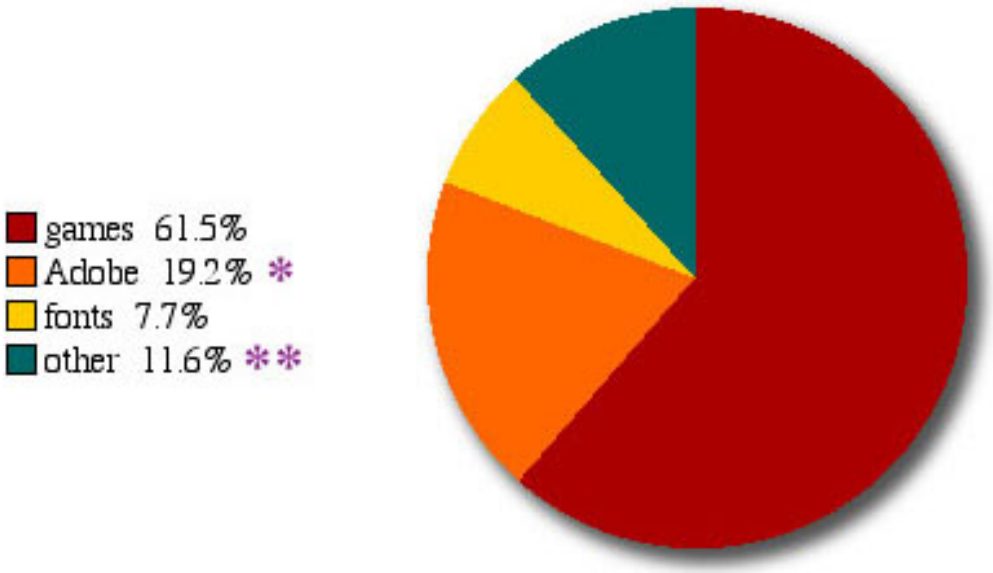
** an Arnold Schwarzenegger ad

The two movies were "The Matrix" and "Scary Movie."



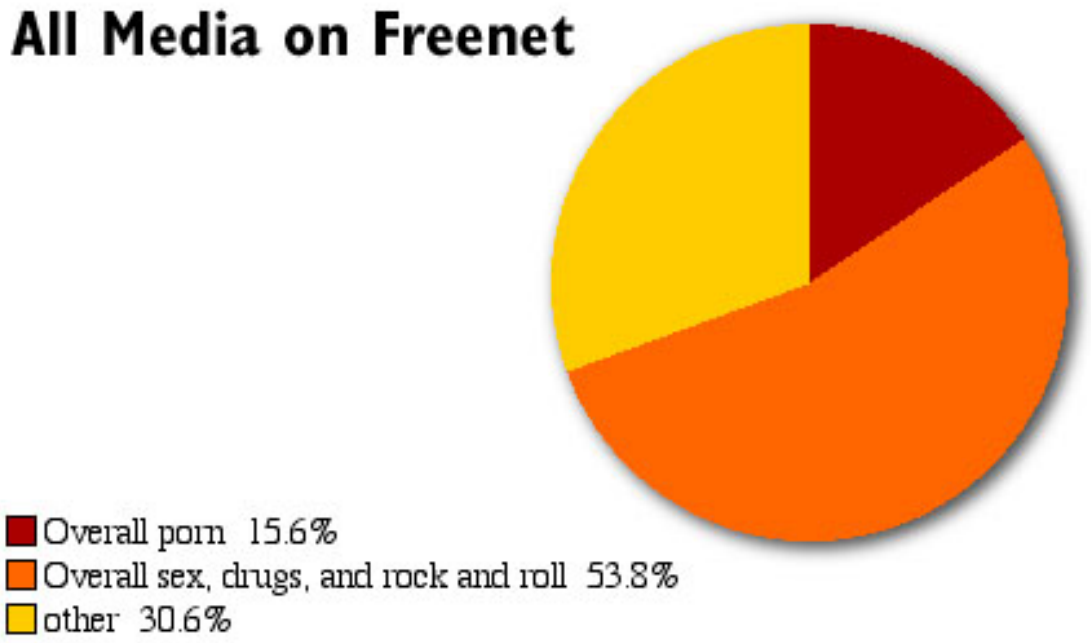
* including two C programs, and no other positively identifiable source code

Of the proprietary software:



* software packages
** includes Mathematica

Of the proprietary games, 81.2% were for Nintendo consoles.



Jon Orwant is the CTO of O'Reilly & Associates and is Editor in Chief of The Perl Journal.

Related Articles:

Free Radical: Ian Clark has Big Plans for the Internet

A Directory of Peer-to-Peer Projects

How Ray Ozzie Got His Groove Back

Open Source Roundtable: Free Riding on Gnutella

O'Reilly's Peer-to-Peer Conference

How the Peer-to-Peer Working Group Ought to Be Organized

Return to the P2P DevCenter.