

Software Deployment and Configuration



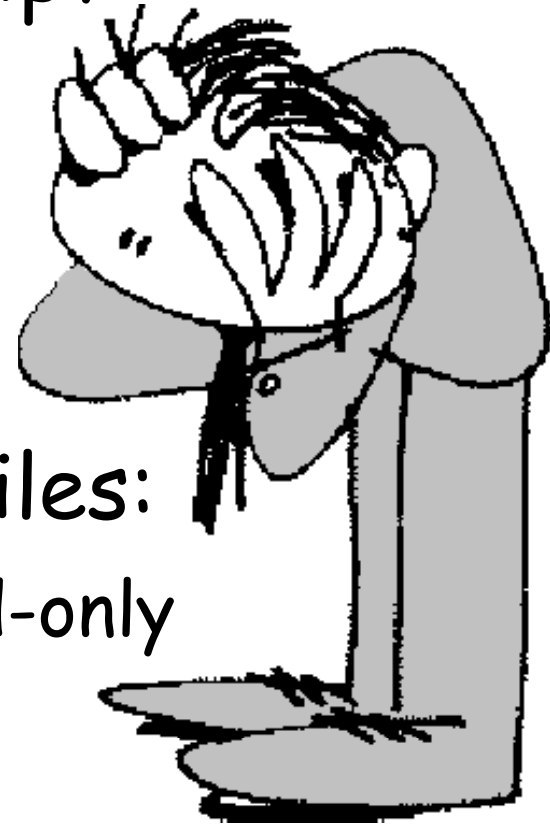
Paul Anderson
Division of Informatics
University of Edinburgh
<paul@dcs.ed.ac.uk>

Configuration And Deployment

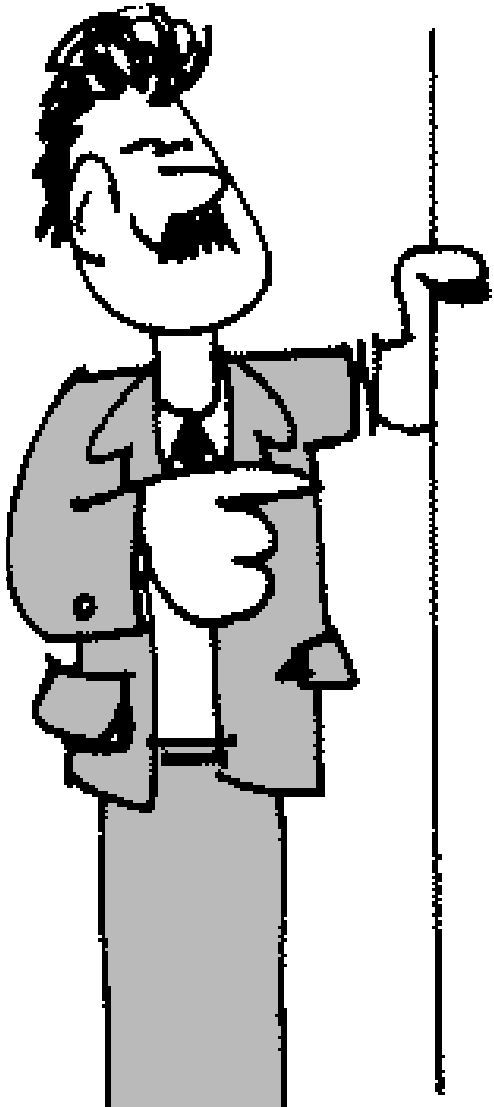
- Configuration (in this context) is the process of customizing an instance of a software package for:
 - A particular site
 - A particular host
 - A particular user (we will not address this)
- Deployment involves the installation of the software, usually on multiple, remote hosts
- Configuration can occur at different stages:
 - Build (compile) time
 - Deployment (install) time
 - Runtime

Nightmare.tgz

- The package has an INSTALL script
 - Runs unknown commands as root
 - Expects an interactive dialog
 - Edits inappropriate system files
 - Installs a daemon
- The package attempts to install files:
 - In a directory which is mounted read-only
 - For an inappropriate architecture
 - Into an automount point
- There is no way of identifying what has been installed, and no way or removing it



Summary



- Choosing pathnames
- Compile-time configuration
- Packaging
 - Package management tools
 - Creating RPMs
 - Install-time configuration
- Deployment
- Runtime configuration
 - LCFG

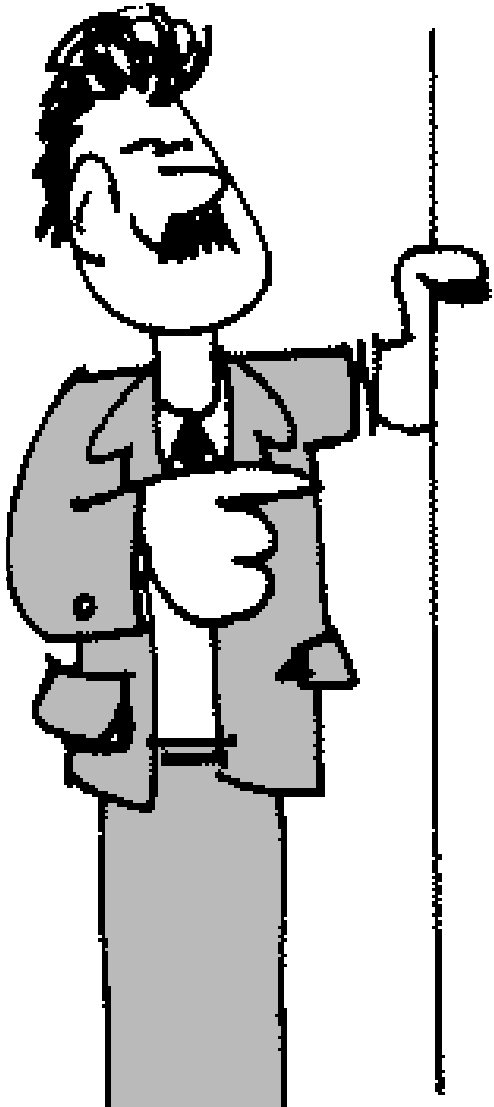
Choosing Pathnames

- Different areas of the filesystem have different properties:
 - Local or network mounted (or an automount point)
 - Single or shared architecture
 - Read-only or read-write
 - Small or large filesystem
- Different pathnames will belong to different areas at different sites
- Pathnames used at install time might not be the same as the pathnames used at runtime!
 - When installing onto a read-only network drive
 - When building RPMS (we don't have root access)

Pathname Standards

- Filesystem Hierarchy Standard:
 - Specifies guiding principles for each area of the filesystem
 - Specifies the minimum files and directories required
 - Enumerates exceptions to the principles
 - Enumerates specific cases where there has been historical conflict
- <http://www.pathname.com/fhs/>
- Individual sites or projects may have their own standards

Summary



- ✓ Choosing pathnames
- Compile-time configuration
- Packaging
 - Package management tools
 - Creating RPMs
 - Install-time configuration
- Deployment
- Runtime configuration
 - LCFG

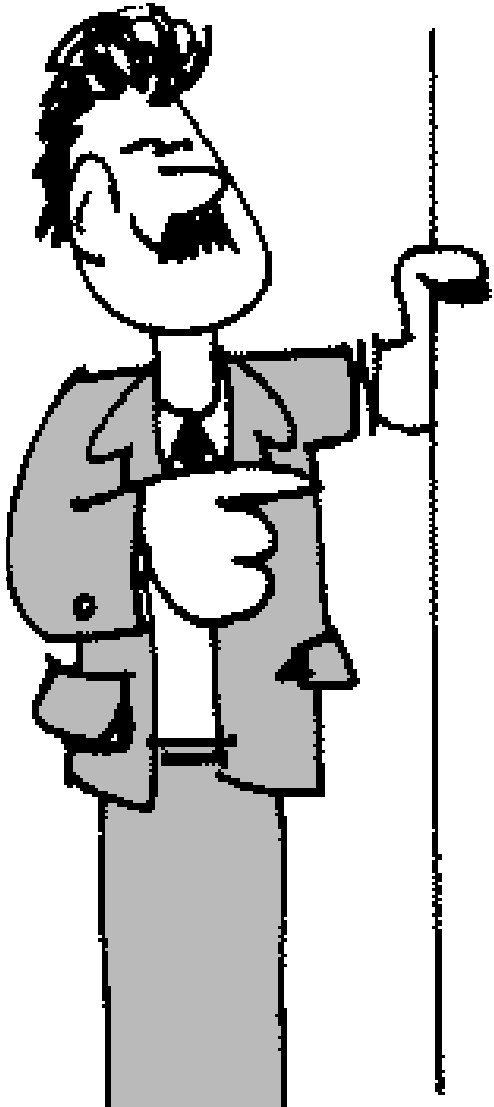
Compile-time Configuration

- It is not always practical to allow everything to be configured dynamically at runtime
 - Eg. Selecting a threading or non-threading library
- Some parameters must be fixed at compile time
- Compile-time configuration leads to multiple (different) versions of the binary package, and care is required to distinguish between these "flavours".
- Configuration should be part of the standard build process

Compile-time Tools

- In simple, cases, storing configuration information in a single header file may be sufficient.
- In a multi-package project, simple ad-hoc scripts might be used to substitute parameters from a common configuration file
 - `www.dice.informatics.ed.ac.uk/doc/dice-buildtools.pdf`
- GNU autoconf is a tool based on an extensible set of m4 macros which can automatically detect many different aspects of the system.
 - Discovered parameters can be used to generate C header files, or substituted in other text files
 - User-supplied parameters can be included
 - `http://www.gnu.org/software/autoconf/`

Summary



- ✓ Choosing pathnames
- ✓ Compile-time configuration
- Packaging
 - Package management tools
 - Creating RPMs
 - Install-time configuration
- Deployment
- Runtime configuration
 - LCFG

Packaging

- A (good) packaging tool supports:
 - Bundling of files into an archive format
 - Recording of filenames to enable de-installation
 - Handling of version numbers to support upgrading
 - File conflict detection
 - Dependency management
 - Run-time dependencies
 - Build-time dependencies
 - Install-time dependencies
 - Execution of pre/post install/de-install scripts

Package Management Issues

- Full benefits are only gained if all the software on a system is handled by the same package management system
 - On many platforms, this is not always possible (and required software may be unavailable in the necessary format)
- Support for “flavours” is not well developed and usually involves encoding in the package filename.
- Some or all files in a package may be “relocatable” so that a pathname prefix can be set at install time

Pre/Post Install Scripts

- Pre/Post install scripts are a frequent source of installation problems. They should be avoided if possible. Otherwise:
 - Ensure that actions taken at install time can be (and are) reversed at de-install time
 - Do not modify files belonging to other packages
 - In particular, do not modify system configuration files - /etc/passwd, /etc/inetd.conf, etc..
 - Do not assume the availability of user-interaction, or even a console device
- If some modifications to the system-wide configuration are necessary it is useful to document these, and/or to provide a separate script.

Package Management Tools

- RPM

- Manages all software packages on Redhat Linux systems
- <http://www.rpm.org/>

- GPT

- The Globus packaging tool
- <http://www-unix.globus.org/packaging/>

- Solaris pkgadd

- http://sunsite.org.uk/solaris_freeware/pkgadd.html

- Conversion between some formats is possible automatically: Eg. GPT => RPM

RPM

- An RPM package is created from:
 - Packed source file (`foo.tgz`)
 - Patches (`foo1.patch`)
 - Spec file (`foo.spec`)
- A single command can build the package:
 - `rpm -ba foo.spec`
- This creates a binary RPM with the architecture as a "flavour" in the filename:
 - `foo-2.35-1.i386.rpm`
- It also creates a source RPM (SRPM) containing everything necessary to rebuild from the source:
 - `foo-2.35-1.src.rpm`

Creating RPMs

- Creating RPMs involves
 - Packaging sources into a tar file
 - Writing a specfile
 - Using `rpm -ba`
- This process should be integrated with the build process
 - It is useful to be able to reconstruct a whole set of RPMs from a CVS repository.
 - This is possible, if each module supports, for example: `"make rpm"`
- Building multiple package formats for the same package may be necessary to support multiple platforms

A Skeleton Specfile

Summary: .. description

Name: foo

Version: 2.35

Release: 2

Source: foo-2.35.tgz

%prep

%setup foo-2.35.tgz

%build

make

%install

make install

%files

/usr/bin/foo

A Real Specfile (1)

Summary: change protection on Zip disk

Name: ziplock

Version: 1

Release: 2

Copyright: GPL

Group: Utilities/System

Source: ziplock-1-2.tgz

Packager: Paul Anderson<paul@dcs.ed.ac.uk>

BuildRoot: /var/tmp/ziplock-build

A Real Specfile (2)

```
%description
```

```
This program .....
```

```
%prep
```

```
%setup ziplock-1.2
```

```
%build
```

```
make
```

A Real Specfile (3)

%install

```
rm -rf $RPM_BUILD_ROOT
```

```
mkdir -p $RPM_BUILD_ROOT/usr/bin
```

```
mkdir -p $RPM_BUILD_ROOT/usr/man/man1
```

```
make install PREFIX=$RPM_BUILD_ROOT
```

%files

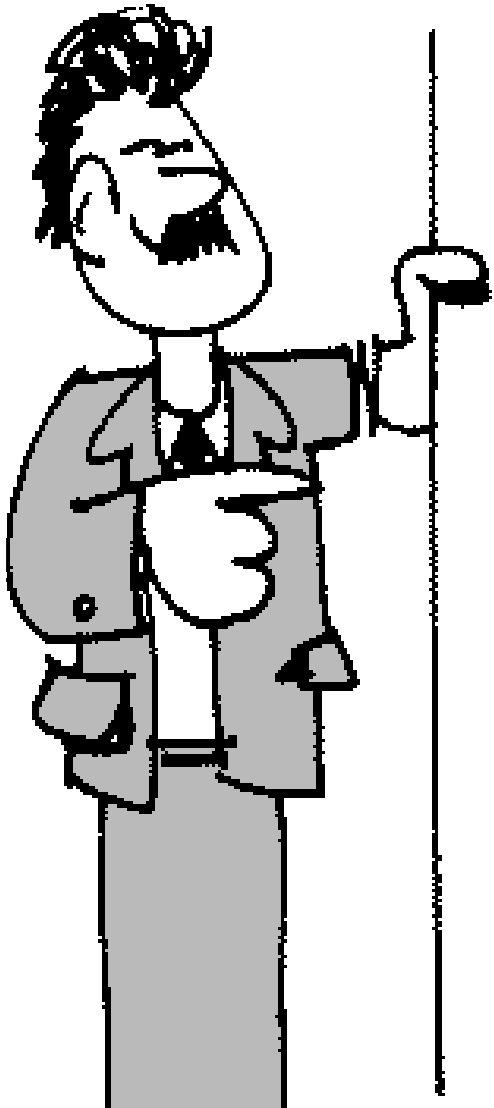
```
%defattr(-,root,root)
```

```
%doc README ChangeLog TODO
```

```
%doc /usr/man/man1/ziplock.1
```

```
/usr/bin/ziplock
```

Summary



- ✓ Choosing pathnames
- ✓ Compile-time configuration
- ✓ Packaging
 - Package management tools
 - Creating RPMs
 - Install-time configuration
- Deployment
- Runtime configuration
 - LCFG

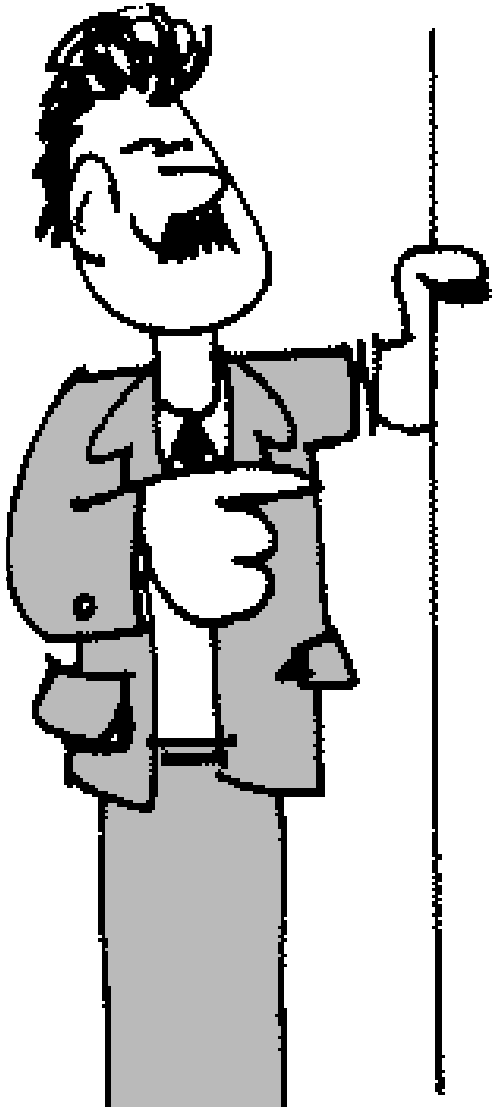
Installation

- A binary RPM can be installed with a single command:
 - `rpm -i foo-2.35-1.i386.rpm`
- This:
 - Validates prerequisite dependencies
 - Checks for file conflicts
 - Executes and pre-installs scripts
 - Installs the files
 - Records the installed files in a database
 - Executes and post-install scripts
- The rpm can later be removed with:
 - `rpm -e foo`

Deployment

- Large-scale deployment tools will manage the packages on a cluster of machines by automatically scheduling installs, de-installs and the correct ordering for updates.
 - Eg. `updaterpms`
 - `www.dcs.ed.ac.uk/home/ajs/linux/updaterpms/index.html`
- The required package sets for each machine are specified in a central configuration file
- Correct dependency information is important
- Some tools will automatically monitor a repository for newer versions

Summary



- ✓ Choosing pathnames
- ✓ Compile-time configuration
- ✓ Packaging
 - Package management tools
 - Creating RPMs
 - Install-time configuration
- ✓ Deployment
- Runtime configuration
 - LCFG

Run-time Configuration

- Runtime configuration is typically performed by reading configuration files
 - (Although, a program might also use external data sources such as LDAP)
- Any configuration files deployed with the code can only be considered as a default because they usually need to contain host-specific information
- Normally, the local site will provide some way of populating these configuration files
- Existing configuration files must not be overwritten when updating an RPM version!

LCFG

- LCFG is a configuration framework developed At Edinburgh University and currently being used by the European DataGRID testbeds
 - www.lcfg.org/
- Site-wide configuration information is specified in a central configuration repository which is compiled into individual host "profiles"
- The XML profiles are distributed to the clients over HTTP
- Scripts on the clients substitute parameters from the profile into individual configuration files

An LCFG Template File

```
# The name of the maildrop file
```

```
mmdflfil: .mail
```

```
# Hardcoded POP server name
```

```
pophost: <%mhpop%>
```

```
# List of smtp servers
```

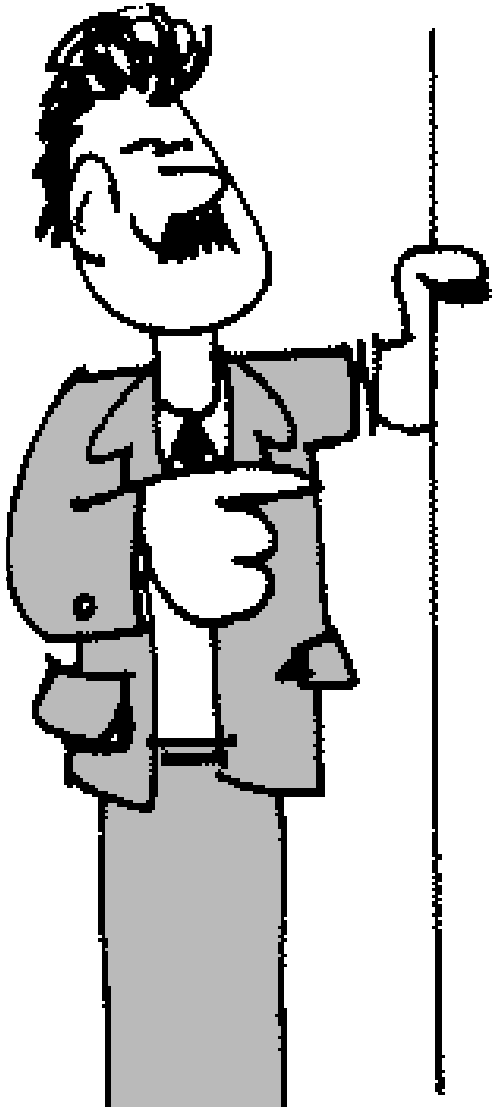
```
<%if:<%mhsntp%>%><%else:%>#<%end:%>
```

```
servers: <%mhsntp%>
```

Dynamic Reconfiguration

- LCFG will normally update configuration files as soon as a central configuration change occurs
- Daemons must be prepared to reconfigure "on-the-fly" wherever possible, either on receipt of a signal, or by monitoring the configuration file for changes
- Programs can access LCFG configuration information directly

Summary



- ✓ Choosing pathnames
- ✓ Compile-time configuration
- ✓ Packaging
 - Package management tools
 - Creating RPMs
 - Install-time configuration
- ✓ Deployment
- ✓ Runtime configuration
 - LCFG

Some Key Points

- Use (global or local) standard pathnames, but make them configurable
- Integrate compile-time configuration and package construction with the build system
- Distribute software in a standard package format
- Avoid intrusive install scripts
 - If changes are required to other parts of the system, allow the system manager flexibility in how this is achieved
- Be prepared to reconfigure long-running processes "on-the-fly"