

# Autodir HOWTO

**Venkata Ramana Enaganti**

<ramana <> intraperson dot com>

2004-09-23

## **Revision History**

Revision 1.00	2004-09-23	Revised by: VRE
Initial release, reviewed by Rahul Sundaram at TLDP		
Revision 0.32	2004-09-13	Revised by: VRE
New sections like requirements and others.		
Revision 0.10	2004-06-24	Revised by: VRE
second draft		
Revision 0.9	2004-06-11	Revised by: VRE
first draft		

This HOWTO is about *Autodir* installation, configuration and other issues related to *Autodir*.

---

# Table of Contents

<b>1. Introduction</b> .....	<b>1</b>
<u>1.1. Copyright and License</u> .....	1
<u>1.2. Disclaimer</u> .....	1
<u>1.3. Feedback</u> .....	1
<u>1.4. New Versions of this Document</u> .....	1
<u>1.5. Credits / Contributors</u> .....	1
<b>2. Before going to details</b> .....	<b>2</b>
<b>3. Why not pam mkhomedir?</b> .....	<b>3</b>
<b>4. Where it can be used</b> .....	<b>4</b>
<b>5. What it is not</b> .....	<b>5</b>
<b>6. Differences between Autodir and Autofs</b> .....	<b>6</b>
<b>7. How it works</b> .....	<b>7</b>
<b>8. Some definitions</b> .....	<b>9</b>
<b>9. Directory organization under real base directory</b> .....	<b>10</b>
<b>10. Virtual directory expiration</b> .....	<b>11</b>
<b>11. Backup support</b> .....	<b>12</b>
<b>12. Backup program requirements</b> .....	<b>13</b>
<b>13. Module options</b> .....	<b>14</b>
<b>14. Autodir requirements</b> .....	<b>15</b>
<b>15. Autofs kernel module</b> .....	<b>16</b>
<b>16. Importing user and group accounts</b> .....	<b>17</b>
<b>17. Getting it</b> .....	<b>18</b>
<b>18. Managing Home directories</b> .....	<b>19</b>
<u>18.1. Base directories for autohome</u> .....	19
<u>18.2. Directory organization</u> .....	20
<u>18.3. Misc suboptions for autohome</u> .....	20
<u>18.4. Summing up with an example</u> .....	20
<b>19. Managing group directories</b> .....	<b>21</b>

# Table of Contents

<b><u>20. Autodir options</u></b> .....	<b>22</b>
<b><u>21. Backup options</u></b> .....	<b>23</b>
<b><u>22. Examples</u></b> .....	<b>24</b>
<b><u>23. RPM specific</u></b> .....	<b>25</b>
<b><u>24. Further Information</u></b> .....	<b>26</b>

# 1. Introduction

*Autodir* offers a simple and effective means to create directories like home directories in a transparent manner. It relies on the autofs protocol for its operation.

This document explains how to create directories on demand using *Autodir* in a transparent way to the applications. This document also explains using transparent backup feature that is possible with *Autodir* without bringing system down for backup purpose for all directories managed by *Autodir*.

---

## 1.1. Copyright and License

This document, *Autodir HOWTO*, is copyrighted (c) 2004 by *Venkata Ramana Enaganti*. This work is licensed under the Creative Commons Attribution License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/2.0/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

Linux is a registered trademark of Linus Torvalds.

---

## 1.2. Disclaimer

No liability for the contents of this document can be accepted. Use the concepts, examples and information at your own risk. There may be errors and inaccuracies, that could be damaging to your system. Proceed with caution, and although this is highly unlikely, the author(s) do not take any responsibility.

All copyrights are held by their by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark. Naming of particular products or brands should not be seen as endorsements.

---

## 1.3. Feedback

Feedback is most certainly welcome for this document. Send your additions, comments and criticisms to the following email address : <[ramana <> intraperson dot com](mailto:ramana@intraperson.com)>.

---

## 1.4. New Versions of this Document

The latest version of this HOWTO will be made available from [here](#).

---

## 1.5. Credits / Contributors

In this document, I have the pleasure of acknowledging for language and technical review work:

- Rahul Sundaram<[rahulsundaram@yahoo.co.in](mailto:rahulsundaram@yahoo.co.in)>
-

## 2. Before going to details...

After releasing intraperson beta, I started working on a administration guide that deals with administration aspects of *intraPerson*. For more details check <http://www.intraperson.com>. But I was stuck with one simple thing. It is easy to create users in ldap -- at least I think so; but how to create home directories for those users in ldap wherever those ldap accounts are imported?

I found some solutions But I was not satisfied as every solution has serious drawback attached with it. But after going through autofs documents and hacking little bit, I arrived at conclusion that autofs protocol may offer much better solution to this challenge.

The result is *Autodir*, based on the autofs protocol.

---

### 3. Why not pam\_mkhome?

The PAM module `pam_mkhome` uses Pluggable Authentication Module architecture for its operation. As such, there are some limitations associated with it. For instance:

- Some servers may not authenticate users but they may expect user directories to exist. This means they do not use PAM, and in turn, `pam_mkhome` does not get a chance to create home directories. The notorious example is on email servers.
- PAM is always an optional component for authentication. Some may not use PAM at all and use a different method to authenticate users. In this case `pam_mkhome` is never going to be used.
- Generally `/home` is owned by root and only root users can create home directories in it. Therefore the server that wishes to create home directories through PAM must be run as root, or else the home directory must be made similar in permission to `/tmp`.

Finally, *Autodir* is much wider in scope and supports many more interesting features.

---

## 4. Where it can be used

- Where user accounts reside in centralized database like ldap, SQL, NIS, NIS+ or other databases, from which user and groups are imported to other systems. To create, for example home, group directories in those systems which import these accounts from centralized database, on demand.
  - To exploit its transparent backup feature for 24\*7 online systems.
  - It can be also used even when accounts are in a local system, to some extent hiding what accounts exist in /home directory, for example.
-

## 5. What it is not

*Autodir* can create directories but it does not remove them once user, group entries are removed from system accounts database. And there may be some more limitations with modules used with *Autodir*. Check appropriate sections.

---

## 6. Differences between Autodir and Autofs

Now the important issue arises as there is already an autofs package to handle mounts and Autodir is in similar line with the autofs package.

- The main purpose of autofs is to deal with network mounts on demand instead of mounting all at the same time, which results in preserving system resources. Though there is some support in the autofs package to mount home directories on demand, the requirement is that *these home directories must exist already*.

On the other side, *Autodir* specializes *only* in local directory creation and mounting them on demand.

*Autodir* can also create real directories in disk file systems such that they do not reside in one single flat base directory. This is how utilities like **useradd** create by default. In a standard file system setup, all home directories reside in base `/home` directory. For file systems like ext2, ext3 performance will degrade if large number of home directories exist in single base directory.

For applications accessing these directories, *Autodir* presents all directories for them in a *single* autofs mounted virtual base directory *on demand*; actual directories are created in subdirectories of some other directory in hierarchical style.

For example, the real home for a user with uid `user1` will be created as `/autohome/u/us/user1` if configured that way, but mounted in `/home` on demand for applications accessing home directory in `/home/user1`.

Permissions for real base directory, where actual home directories are kept `/autohome` in the above example, are kept in such a way that `/autohome` can not be accessed by anyone except by root.

This mounting of directories on demand and unmounting when not in use presents an interesting opportunity — the ability to tell when a directory is in use and when it is not in use. This simply means a program like backup can be started when a directory is unmounted.

*Autodir* exploits this capability by starting the command–line mentioned backup whenever a directory becomes unused.

- There is one more important issue to be presented if you are an administrator reading this document. *Autodir* does not call external programs **mount** and **umount**, as is the case with the autofs package; rather, it uses system calls directly. As a side effect, it is faster and more reliable, but `mtab` is not updated. I felt this was not necessary as all mounts and unmounts are local directories.
  - Another minor difference is that *Autodir* is completely *multi–threaded*. Autofs is also expected to be multi–threaded in future versions.
-

# 7. How it works

*Autodir* uses modules to get specific functionality. The core *Autodir* implements generic functionality on which modules can exploit and add specific functionality of their own.

At any moment only one module can be added to *Autodir*. If there are two modules, for example `autohome`, `autogroup`, two processes of *Autodir* should be created so that each process will have required modules attached to it.

For further explanation I chose the `autohome` module which handles transparent home directory creation.



- `autohome` module creates user home directories on demand if these does not exist already.
- It is assumed user accounts exists but not their home directories. Either because these accounts were created with the `-M` option with **useradd** or these accounts were imported from ldap, NIS or some other external database for which home directories are yet to be created.
- It also assumed *for this explanation only* that all user home directories are expected to be in the `/home` directory.

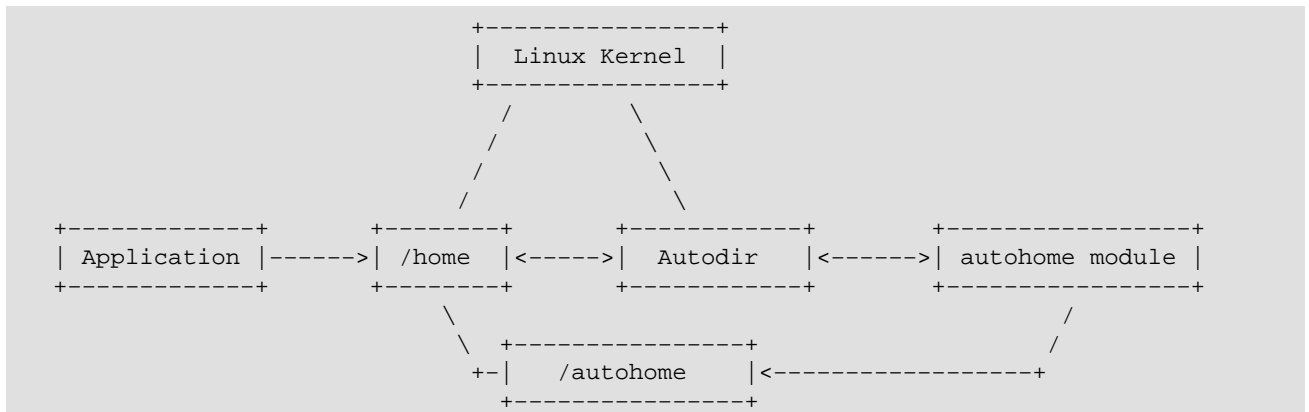


Some fine details are intentionally kept aside to make explanation easy to understand.

First `autofs` file system is mounted on `/home` directory by *Autodir*. And this is informed to the Linux kernel that `/home` is managed by user space application *Autodir* from now on.



Do not bother too much about `autofs` file system if you do not understand about it. Just think some special kind of file system something in similarity with memory based file system but with some additional special properties.



Whenever an application or daemon needs access to user's home directory, for example `/home/userhome1`, they directly enter into `/home/userhome1` to access it. Kernel which notices this, informs to *Autodir* if `userhome1` directory does not exist already in `/home`.

*Autodir*, in turn, passes this request to `autohome` module. `autohome` module does not touch `/home` directory. Instead it manages *real home directories* some where else, for example in `/autohome` as shown in

## Autodir HOWTO

the above figure.

`autohome` module creates real home directory if it does not exist already in `/autohome` directory. After it is successfully created or failed to created, whatever the outcome, it is reported back to *Autodir* along with the path to real home directory -- if successful.

If `autohome` module reports success, *Autodir* creates `userhome1` directory under `/home` and mounts *real home directory* from `/autohome` on it. At the end *Autodir* informs this to the kernel whether this whole operation successful or failure. Accordingly kernel allows application to enter the directory or reports that no such directory exists, in case of failure, back to the application.

---

## 8. Some definitions

Before going further it is better to understand the following terms to simplify explanation.

### *Virtual directories*

These directories do not exist on disk. Instead these are created and deleted on demand in memory. If system reboots all these directories vanish. In the previous figure, all directories under /home are *virtual directories*.

### *Virtual base directory*

This is the directory that holds all *Virtual directories*. This directory *does* exist on disk and therefore it remains even after reboot. In the previous figure /home is *virtual base directory*.

### *Real directories*

These are the directories that actually reside on the disk. Even after reboot, these remain intact. In the previous figure all directories created under /autohome are *real directories*.


### *Real base directory*

This is the directory that holds all real directories. In the above figure /autohome is *real base directory*.

Each *virtual directory* is mapped to *real directory*. Which means whatever written or modified to *virtual directory* is actually sent to *real directory*.

On reboot of the system *real directories* and their content remain intact. But *virtual directories* are again created on demand as exactly as they were before.

*Virtual directories* are removed if these are not used for a specified time period and created again if necessary. When *Virtual directory* is removed backup program is started on corresponding *real directory* if backup is configured.

 Applications should access only *virtual directories*. *Real directories* are hidden from applications except for root. But there is one exception. Backup programs always access only *real directories*.

---

## 9. Directory organization under real base directory

Why special organization under *real base directory*? If we just create all *real directories* in one *real base directory* there could be performance penalty when there are large number of *real directories* to be created. File systems like ext2/ext3 are not optimized for this kind of flat directory structure.

It would be much better if *real base directory* is divided into more subdirectories or even dividing these subdirectories again into more subdirectories. And in the final subdirectories actual home directories are kept!

There are three types of directory organization.

*level 0*

Actually no organization. All home directories are created directly under *real base directory*.

*level 1*

*Real base directory* is divided into more subdirectories. These subdirectories names are derived from first letter of the final directory to be created. For example, if `user1` directory is to be created, first a directory named 'u' is created under *real base directory*. Then in that subdirectory actual directory `user1` created as `<real_base_directory>/u/user1`.

*level 2*

Same as level 1 organization but after first level of subdirectories, second level subdirectories also created. Name for which is derived from starting two letters of the final directory to be created. For example, for user `user1` as with the above example, `<real_base_directory>/u/us/user1` is created.

---

## 10. Virtual directory expiration

When an application tries to access *virtual directory* in *virtual base directory*, *Autodir* creates *virtual directory* in it if it does not exist already and mounts the *real directory* on it from *real base directory*. But once this happens and if this *virtual directory* is not accessed from *virtual base directory* for a specified time period by any application, this directory is removed and accordingly that corresponding *real directory* in *real base directory* is marked for backup.

The time period to wait for expiration can be given through command line option to *Autodir*.

---


# 11. Backup support


*Autodir* supports backup program launching when a *virtual directory* is removed after a period of inactivity. Removal of *virtual directory* is itself an assurance that no other application can access the content and modify it.


Like there is wait duration for expiring *virtual directory*, for backup also *Autodir* waits some more time, after *virtual directory* expiration, for starting backup. This time period can be configured through command line option to *Autodir*.

By design, backup programs are expected to operate on *real directory* but not on *virtual directory*. If backup program try to access *virtual directory* *Autodir* assumes some regular application is in need of that directory and backup program is killed even if the *virtual directory* accessing process is backup program itself.

A separate backup process for each *real directory* is used. The backup program can be given arguments of *real directory* on which to operate.

 Backup support is independent of any particular module being used. It is applicable to all modules with *Autodir*.


 Backup programs should never access *virtual directory* or *virtual base directory*.


 Backup feature is not much useful if *virtual directories* are accessed all the time by applications.

---

## 12. Backup program requirements

*Autodir* demands some extra requirements from backup program being used. The reason for this is that when backup is working on *real directory* and with corresponding expired *virtual directory* and that *virtual directory* is requested again by an application while backup is running, backup is killed. First SIGTERM is sent to gracefully stop it. But if it does not shutdown in time — one second at this moment; SIGKILL will be sent which is guaranteed to stop the backup.

 When and only when backup stopped, application is given access to the *virtual directory* requested.

 Whatever backup is used, it should be able to recover from this signal gracefully, not causing unrecoverable side effects.

One more important issue is that the environment under which it is run. All backup programs are run as root user. But at the same time all unnecessary root privileges are taken away using POSIX capabilities. In other words these backup programs can read any file or directory that belongs to any user on the system and nothing more than that. Other than that it is like ordinary user level process.

---

## 13. Module options

There are two kinds of options that can be passed to *Autodir*. In the first type, options are for **autodir** itself and are common irrespective of which module is used. There are other type of options which are specific to the module being used. These options called suboptions and are passed to the module being used differently with main option `-o`. This is similar to **mount** command suboptions.

For example, suboptions to the example module `autohome` can be passed as,

```
-o 'realpath=/tmp/autohome,level=2,noskel'
```

Here `realpath`, `level`, `noskel` are suboptions for `autohome` module.

---

## 14. Autodir requirements

- Linux kernel equal to or later version of 2.4. These kernel versions support mounting one directory on another directory. At this moment *Autodir* is not ported to other Unices but this may change in future.
  - *Autodir* requires autofs kernel module based on protocol version 3. But it does not require autofs user level package. Autofs kernel module is pretty standard and almost all distributions include it.
-

## 15. Autofs kernel module

*Autodir* uses autofs kernel module for its operation. Kernel module `autofs` must be loaded before even starting **autodir** for its proper operation.

This can be done as root user and using **modprobe** command as follows,

```
# modprobe autofs
```

---

## 16. Importing user and group accounts

If user and group accounts reside in centralized database these must be imported before starting *Autodir*. How to do this is out of scope of this HOWTO. But there are number of documents which explain how to do this in clear manner.

---

# 17. Getting it

At this moment *Autodir* available in tar, rpm formats. More information can be found at <http://www.intraperson.com/autodir/>.

If source is downloaded, follow these simple steps to install it.

- Unpack the source.

```
$ tar zxvf <tar file name>
```

- Move to the expanded directory and execute the following.

```
$ ./configure
```

```
$ make
```

```
# make install
```







configurescript check for required libraries. If these are not present it will stop from proceeding.

---

# 18. Managing Home directories

This section will explain how to configure *Autodir* so that user home directories are created on demand. For this purpose *autohome* module is used which deals with specifics of home directory creation.

To load *autohome* module with *Autodir*, use option `-m`. For example, `-m /usr/lib/autodir/autohome.so`.

-  When an application tries to access home directory, that home directory is used to check if there is any user with user name same as the directory being accessed. If user name exist with this criteria then home directory is created. Otherwise no such file or directory is reported back to application.
-  *autohome* does not deal with creating user accounts on local systems or in ldap or in any other database. It only deals with creating home directories once these accounts exist and imported to local system from databases like ldap, NIS.
-  It is worth mentioning one limitation with *autohome* module. It expects that user name and home directory are related to each other. For example, for user `user1` the home directory should be `/home/user1` or `/some/directory/name/user1` but not `/some/directory/name/userhome1`. This can be supported but it will be burden on system resources as each password entry has to be examined from first to last.
-  If the existing user password database is such that user home directories are distributed under different base directories, for example `/home/class1/user1`, `/home/class2/user2332`, then *autohome* configuration becomes complicated and it is not recommended.

---

## 18.1. Base directories for autohome

Next step in setup is to decide where will be *virtual base directory* and *real base directory* for home directory creation.

What is *virtual base directory* and what is *real base directory* in the context of *autohome* module?

It all depends on how user accounts are created. If an user account created for user name `user1` with home directory `/home/user1` then `/home` will become *Virtual Base Directory*.

Then what is *real base directory*? It can be any directory. Only thing that has to be kept in mind is, there should be enough space as all actual files are stored here instead of in *virtual base directory*.

In most server configurations `/home` is a separate partition mounted on it. But if `/home` is made *virtual base directory* files are not stored in that directory! The solution is, do not mount partition on `/home` but instead mount it under somewhere else and make it *real base directory*.

*Autodir* option `-d` is used to specify *virtual base directory*. For example `autodir -d /home` assuming `/home` is *virtual base directory*.

It is little tricky to specify *real base directory*. *real base directory* is managed by *autohome* module so this option must be passed to the module through module suboptions. If the *real base directory* is

`/var/autohome` then it is specified with option `-o` as `-o realpath=/var/autohome`.

---

## 18.2. Directory organization

Please refer to [directory organization under real base directory](#) for detailed explanation of this topic.

`autohome` does support this kind of organization. The suboption used to specify directory organization desired, is with `level` suboption. For example, `-o level=2`.

---

## 18.3. Misc suboptions for autohome

Suboption `skel` can be used if skeleton path is not default value `/etc/skel` like `-o skel=/some/other/dir`.

Suboption `noskel` can be used with `-o` to indicate not to copy skeleton files to home directories when created.

---

## 18.4. Summing up with an example

First, import user accounts from centralized database like, for example, `ldap`.

Next, `autofs` module must be loaded. This can be done as described in [autofs kernel module section](#).

If `/home` is to be used for home directories then `/home` will become *virtual directory* and specified to **autodir** with `-d /home` option.

Assuming `autohome` module is located at `/usr/lib/autodir/autohome.so`, this module can be loaded with **autodir** as `-m /usr/lib/autodir/autohome.so`. Note that full path for module is given.

Where actually real home directories reside is given with `realpath` suboption. If it is `/autohome`, it can be given as `realpath=/autohome`.

With all these options **autodir** can be started as,

```
# autodir -d /home \
        -m /usr/lib/autodir/autohome.so \
        -o 'realpath=/autohome'
```


Once *Autodir* is started, `/home` directory will be blank in the beginning. Whether *Autodir* working properly or not can be tested by changing directory to one of the home directories as root user or as the owner of the home directory.

---

# 19. Managing group directories

`autogroup` module is for creating directories on demand for common group access. It can be used with Samba, for example, to dynamically create shared directories for group of people.

 `autogroup` module check for requested directory with valid groups from system group database.

 `autogroup` can be used to create home directories as well! Provided that there exists user private group for each user. This way all group and home directories can be created at one place with one module. But no skeleton files are copied and the `autogroup` suboption `nopriv` should not be used.

`autogroup` configuration is same as `autohome` module but unlike `autohome`, *virtual base directory* can be placed anywhere and any name can be given to it. It is not dictated by system accounts.

The module `autogroup` can be used with *Autodir* using option `-m`. For example, `-m /usr/lib/autodir/autogroup.so`.

All suboptions explained in [managing home directories](#) are same for `autogroup` except `skel`, `noskel` as these are meaningless for `autogroup` module. But there are two other suboptions specific for `autogroup`. These are given below.

*nopriv*

Some Linux installations use user private groups. If directories for these groups are not to be created, then use this suboption.

*nosetgid*

By default `setgid` is set on group directories created. Use this suboption to disable this feature.

---

## 20. Autodir options

In this section some of the options to *Autodir* are explained. Backup options are explained in [backup section](#).

- `-d` For specifying *virtual base directory*. If this path does not exist, it will be created. Absolute path is expected for this option.
  - `-t` Expiration timeout for *virtual directories*. For more details refer to [virtual directory expiration](#).
  - `-m` Module to be used with *Autodir*. Currently `autohome` and `autogroup` are available. Full path to the module expected.
  - `-o` All options that are to be passed to module are given here. This option passing syntax is similar to **mount** command with its `-o` option. See specific module sections for more info.
  - `-f` Stay foreground and log all messages to the console. For debugging purpose and to see how *Autodir* works.
  - `-l` This option expects path name to filename to which *Autodir* will write its process id.
  - `-h` Help about all options supported.
  - `-v` Version information about *Autodir*.
-

## 21. Backup options

These options are passed to *Autodir* to request backup services.

`-b`

This is the main option to specify backup program path and arguments to it. The path given should be absolute path otherwise *Autodir* does not accept it.

`-w`

Whenever a *virtual directory* is not used for a period of time, it is assumed inactive and it is unmounted. After unmounting directory, whether to launch backup immediately or to wait some more time is decided with this option. It takes arguments in seconds. It is the *minimum* time to wait before starting backup after *virtual directory* expiration. It should not exceed more than one day.

`-P`

This is the priority to be given to backup process. *This is in the range of 1 to 40 inclusive*. Lower value mean higher priority and vice versa. Default value is 30.

`-c`

This restricts maximum number of backup process at any given time. Default is 150.



Argument for `-b` is inclusive of absolute backup program path as well as its own arguments. Therefore it is recommended to use single quotes around this argument

Option `-b` takes path to executable file as well as arguments to it. But the arguments to it are interpreted for `%x` character sequences and replaced with predefined strings as follows.

`%N`

Replaced with *virtual directory* name.

`%L`

Replaced with absolute path to *real directory*.

`%K`

Replaced with host name.

*Others*

Others are fed to `strftime`. See man page for `strftime` for more information.

---

## 22. Examples

```
# autodir -d /home \
    -m /usr/lib/autodir/autohome.so \
    -t 1 \
    -f \
    -o 'realpath=/autohome,level=1,skel=/etc/skel' \
    -l /var/lock/autodir
# autodir -d /home \
    -m /usr/lib/autodir/autohome.so \
    -t 300 \
    -b '/bin/tar cf /tmp/%N%F.tar %L' \
    -w 600 \
    -o 'realpath=/tmp/autohome,level=2,noskel' \
    -l /var/lock/autodir
# autodir -d /var/abase/ \
    -m /usr/lib/autodir/autogroup.so \
    -t 300 \
    -b '/bin/tar cf /tmp/%N%F.tar %L' \
    -w 86400 \
    -o 'nopriv,nosetgid,realpath=/var/realbase,level=0'
```

## 23. RPM specific

*Autodir* can be installed from rpms as,

```
# rpm -ivh autodir-0.28-4.i386.rpm
```

When installed from rpms, two startup scripts are provided namely `/etc/rc.d/init.d/autohome` and `/etc/rc.d/init.d/autogroup`. One for starting *Autodir* with autohome module and another for starting with autogroup module.

Script configuration files `/etc/sysconfig/autohome`, `/etc/sysconfig/autogroup` can be used to specify what options can be passed to *Autodir*.

---

## 24. Further Information

*Mailing list for autodir* <http://lists.sourceforge.net/mailman/listinfo/intraperson-autodir>.

Official website is at <http://www.intraperson.com/autodir/>.

Autofs mailing list <http://linux.kernel.org/mailman/listinfo/autofs>.

Automount HOWTO can be found at <http://www.tldp.org>

Autofs Hacking <http://www.goop.org/~jeremy/autofs>.