GNU Finger

Edition 1.3, for GNU Finger version 1.37
October 1992

by Jan Brittenson and Brian Fox

Introduction

GNU Finger is a utility program designed to allow users of Unix hosts on the Internet network to get information about each other. It is a direct replacement for the Berkeley 4.3 finger code, although it produces different looking output and is designed to run on a wide variety of systems.

Why Another Finger?

Originally, each host on the Internet network consisted of a single, reasonably powerful computer, capable of handling many users at the same time. Typically, a *site* (physical location of computer users) would have only one or two computers, even if they had 20 or more people who used them. If a user at site A wanted to know about users logged on at site B, a simple program could be invoked to query the host at site B about the users which were logged on.

With the onset of more-power-per-person computing, the mainframe has been set aside. A modern computing facility usually consists of one user per host, and many hosts per site. This makes it a trial to find out about logged on users at another site, since you must query each host to find out about the single user who is logged on. If the site had 20 hosts, you would have to invoke a finger program 20 times just to find out who was logged on!

GNU Finger is a simple and effective way around this problem. For sites with many hosts, a single host may be designated as the finger *server* host. This host collects information about who is logged on to other hosts at that site. If a user at site A wants to know about users logged on at site B, only the server host need be queried, instead of each host at that site. This is very convenient.

GNU Finger is a direct replacement for existing finger programs. Since the finger *protocol* (rules for communication) is very simple, GNU Finger follows that protocol in responding to simple requests. But GNU Finger also implements another protocol which allows two finger programs to exchange information in a predetermined way, which allows faster and wider bandwidth communication.

Finger delivers information about users in varying formats, depending on how it is invoked. `finger` invoked without any options performs a *site wide* finger request, no matter which machine it has been invoked from. Switch arguments exist for getting the “long” form of finger information and for getting information only about the local machine.

If a user on host A wants to know about a user on host B, finger must make a network *connection* to host B. If host B is running a finger program, that program is asked to relay information about the user in question through the connection back to host A, where finger can display it.

GNU Finger also runs a server *daemon* process on the *server* host, whose job is to keep track of which users are logged in to local machines.

An optional and currently unsupported feature is passing of graphic images. This is built on the new protocol. A user at site A (e.g. MIT) may see the picture of a user at site B (e.g. UCSB), by typing a finger request. The conversion of graphic data from one format to another is done through GNU Finger; no site need know where or how such images are stored on any other site to be able to display those images. You should ask your system administrator to find out whether he has chose to include this functionality on your network.

1 Using Finger

1.1 Arguments to GNU Finger

The basic argument to Finger is a *user@host* pair. The *user* portion is the name of the user about whom you would like information. The *host* is a machine that the user has an account on. When invoked in this manner, GNU Finger displays the list of hosts that this user is currently logged in on, or, if the user is not logged in, the last time and location that he or she was. *host* may be expressed as any valid Internet address (i.e. *dot-notation*, *host.domain*, etc).

If *host* is non-existent, the local host is assumed. If *user* is blank or unspecified, it is assumed that you want information about all users.

The content and format of the output of GNU `finger` depends on what is being fingered:

User Displays login information about *user*. If '`--info`' or '`--l`' is also specified, `finger` will display the full name, home directory, shell, mail forwarding, and `.plan` and/or `.project` file. This is what the output will look like:

```
bash$ finger --info bson@gnu.ai.mit.edu
[apple-gunkies.gnu.ai.mit.edu]

Jan Brittenson (bson)
Home: /home/fsf/bson
Shell: /usr/local/bin/bash
Mail forwarded to bson@ai.mit.edu.
No mail.
 User Real Name What Idle TTY Host Console Location
bson Jan Brittenson fgrep *p0 apple-gu (nutrimat.gnu.ai.)
bson Jan Brittenson 1:57 *sb nutrimat

Plan:
 To hack GNU Finger
```

The following is output, in the order listed, when asking for long information ('`-l`' or '`--info`') about a particular user:

- Real and login names.
- Home directory.
- Login shell.
- Mail forwarding.
- Whether the user has any unread mail, and if so, when it was last read.
- Current login information, in the same format as produced by a short finger (see below for an explanation). If the user isn't currently logged on, then the last login time and remote host (if known) is reported.
- A `~/.plan` file. If the file isn't readable by everyone, then a message is printed to this effect.
- A `~/.projects` file. This file, like `~/.plan`, should be readable by everyone.

```
bash$ finger bson@gnu.ai.mit.edu
[apple-gunkies.gnu.ai.mit.edu]
 User Real Name What Idle TTY Host Console Location
bson Jan Brittenson fgrep *p0 apple-gu (nutrimat.gnu.ai.)
bson Jan Brittenson 1:57 *sb nutrimat
```

Here is an explanation of what each column contains in the short example:

`User` The user login name.

`Real Name` The real name of the user.

`What` The current or last program run by the user, depending on the system in use. On System V Release 4, for instance, the current program is shown, but on BSD it will be the last terminated program.

`Idle` The time the user has been idle, as *hours*:*minutes*, or the first 7 characters of a string such as "14 days".

`TTY` The significant portion of the user's terminal connection. Exactly what portion this is is system dependent. For example, on System V Release 4, it might be '`*40`' for '`/dev/pty/40`', while on BSD it might be '`*p0`' for '`/dev/ttyp0`'. If preceded by an asterisk ("*"), then the user allows anyone to send messages to this particular terminal.

`Host` The host the user is logged onto.

`Console Location`
Where the user's console is located. If logged in over the network, then this will be the most significant portion of the remote host name if known. A host name is always parenthesized.

In the general short output (i.e. not for a particular host), GNU Finger lists the least idle login for a particular user on each host. A single user often has several logins, since on some systems each window opened creates its own login record. In addition to the least idle login, the console login is also always listed, regardless of how long it has been idle.

To list *all* login records for a particular, host, use the special target '`.local`'. For example, while the following may be part of the general short finger listing for the host '`mole.gnu.ai.mit.edu`':

```
brendan Brendan Kehoe 5:09 *p8 mole (lisa.cygnus.com)
info InfoMaster 12:12 *p4 mole (hal)
law Jeffrey A. Law 3:52 *p7 mole (128.110.4.17:0.0)
rms Richard Stallman sendmai 1:34 *p1 mole (unix:0.0)
rms Richard Stallman 6 days *co mole
```

The last line is the console. The following might be listed by the command '`finger .local@mole.gnu.ai.mit.edu`':

```
[mole.gnu.ai.mit.edu]
```

```
 User Real Name What Idle TTY Host Console Location
brendan Brendan Kehoe 5:18 *p8 mole (lisa.cygnus.com)
info InfoMaster 12:22 *p4 mole (hal)
law Jeffrey A. Law 4:01 *p7 mole (128.110.4.17:0.0)
rms Richard Stallman sendmai 1:44 *p1 mole (unix:0.0)
rms Richard Stallman 23:08 *p0 mole (unix:0.0)
rms Richard Stallman 1 day, *p2 mole (unix:0.0)
rms Richard Stallman 6 days *co mole
```

Mailing List or Alias

Expands the mailing list or alias and displays the recipients. You always have to use '`--info`' or (or '`-l`') when fingering a mailing list or mail alias, otherwise mail aliases won't be looked up due to the extra processing involved. This is what the output will look like:

```
% finger --info postmaster@gnu.ai.mit.edu
postmaster is an alias for the following:
    Roland McGrath <roland>,
    <tower@prep.ai.mit.edu>,
    Noah Friedman <friedman>,
    Michael I Bushnell <mib>
```

User-defined Target

Allows the remote host to display specific information, such as price lists, literature, or weather forecasts. For example:

```
% finger .site@gnu.ai.mit.edu

This is the FSF GNU Project. For more information, please contact
"postmaster". For information about guest accounts, please contact
"request".
```

1.2 Command Line Options

There are a number of command line options that you can give to GNU Finger:

'`--face`'
'`-f`' Ask Finger for the face of all users information has been requested about. An explicit user list has to be provided.

'`--info`'
'`-l`'
'`-i`' Display the "long" form of information for the users fingered. The exact information returned depends on what finger software is run on the remote host – GNU Finger, for instance, returns specific information. '`-l`' is supplied for backwards compatibility; `finger` as distributed from Berkeley has this option.

'`--brief`'
'`-b`' Display the "short" form of information for the users fingered. This is the opposite of '`--info`'.

‘`--port` *port*’
‘`-P` *port*’ Make a connection to *port*, which can be either a numerical port number or a service name from `/etc/services`.

‘`--help`’
‘`-h`’ Print a description of all options.

1.3 Special User Names

You can give GNU Finger one of several “special” user names. These user names all begin with a period (‘`.`’) and instruct the receiving finger daemon to do something that only a GNU Finger daemon can do. Currently, the “special” names are:

‘`.free`’ Return a list of free machines. ‘Free’ machines are those that have no users logged in, or have been idle for a long time. The information returned makes it clear which one is true.

‘`.all`’ Return the information about every machine that the Finger server knows about. ‘`.all`’ is equivalent to issuing the `finger` command without specifying the user name.

‘`.site`’ Returns information about the site, such as company and location.

‘`.clients`’
Returns a list of the clients that the GNU Finger server knows about. Also lists who is logged onto the console.

‘`.faces`’ Return a list of the faces that this server has available. The last line output tells you how many lines were listed previously.

‘`.local`’ Finger only at the specific machine. This allows finger to continue to be useful even in the event that the server is down. It also allows you to examine *all* the login records of a user. Normally, the server only keeps track of the most recently active login record for each user.

‘`.help`’ Describe services provided by the finger server.

2 Advanced Use

2.1 How Finger Works

GNU Finger is the collective name for a set of programs:

‘`finger`’
: Parses the command line and connects to the finger server, ‘`in.fingerd`’, on the `finger` server. Returns the output from the server. `finger` connects to `in.fingerd` on the host specified in the command line. This is the only program you need to know anything about if you’re a regular user. You should refer to this program as the *finger client* to avoid possible confusion.

‘`fingerd`’
: Regularly connects to `in.cfingerd` on the clients specified in the `fingerdir/clients` file, to obtain finger data. This client data is saved in the file `fingerdir/userdata`. `fingerd` should run on the host specified in the `fingerdir/serverhost` file. `fingerd` should be started at boot time.

‘`in.fingerd`’
: Responds to `finger` connections through `inetd`. Should be attached to the ‘`finger`’ service via `/etc/inetd.conf`. `in.fingerd` behaves somewhat differently depending on what host it runs on: on the server host it reads the `fingerdir/userdata` database, on all other hosts it forwards all requests (unless ‘`.local`’ is the target) to `in.fingerd` on the host specified in `fingerdir/serverhost`.

  `in.fingerd` reads the `fingerdir/userdata` database, various system files, and makes SMTP connections to the host specified in the `fingerdir/mailhost` file.

‘`in.cfingerd`’
: This is the program that responds to call-ins from `fingerd` by sampling the status on the client and forwarding it to `fingerd`. It should be configured to respond to the ‘`cfinger`’ service specified in the `clients` configuration file, or port 2003 if nothing else is specified.

2.2 The `~/.fingerrc` Script

When the GNU Finger server receives a request for information about a user it looks to see if the user has a `.fingerrc` file in the home directory. If such a file exists, and is executable, then this file is executed, and the normal finger output is passed to it as input. Its output becomes what is returned for the request. Thus, it can be used to:

- Disable fingering a specific user by linking `~`*`user`*`/.fingerrc` to `/bin/true`.
- Entirely replace the output of `finger` by ignoring its input. Below is a sample script which could be put in ‘`~price-info/.fingerrc`’.

```
#! /bin/sh
#
# This sample script replaces the output of GNU Finger
#
echo Hack-O-Matic Consulting Services, Inc. "   " `date`
```

```
cat <<ETX

Hi, thanks for asking us about prices on our newly introduced support
services for Free Software. Below is a list of new services; for a list
of our previous services, please send mail to this address and someone
will contact you. Please don't forget to tell us how to reach you.

ETX
cat ~/info/new-stuff
```

- Filter the output to make changes. If the script below is put in '`~bson/.fingerrc`' on the finger server host for the domain '`gnu.ai.mit.edu`':

  ```
  #! /bin/sh
  #
  # This sample filter replaces the "Project:" tag with
  # "Working on:"
  #
  sed -e 's/^Project:$/Working on:/g'
  ```

  Then when '`finger -l bson@gnu.ai.mit.edu`' is run, the output could look something like:

  ```
  Jan Brittenson (bson)
  Home: /home/fsf/bson
  Shell: /usr/local/bin/bash
  No mail.

  Jan Brittenson (bson) is not presently logged in.
  Last seen at wombat.gnu.ai.mit.edu on Tue Sep  1 15:08:12 1992

  No plan.

  Working on:
          1. Hacking GNU Finger
          2. Making friends
  ```

2.3 User-defined Targets

Various special targets can be added as executable files in the directory `fingerdir/targets`. Each file name in this directory starts with a letter describing when to execute it, followed by a hyphen and the target name. The three letters are:

'`l`'	In response to a "long" finger; usually by typing '`finger -l`'.
'`s`'	In response to a "short" finger; usually the default if no options are given.
'`x`'	In response to either a "long" or "short" request.

For instance, the special-target file `x-.help` is run for either '`finger .help`' or '`finger -l .help`', whereas the special- target file `l-prices` is run only for '`finger -l prices`'.

The special-target file is run as super-user, and is given no input. By convention, all GNU Finger sites should support at least:

'`.help`'	Display message describing what features and special targets exist on this site.
'`.site`'	Display message describing the site. This may include such things as the company name, its address, and how to contact the system manager.

2.4 How Finger Picks a Port

When invoked, the GNU Finger client looks to see if a '`--port`' option was specified on the command line. If so, then this becomes the port number or service used. Otherwise, the client looks to see what name it was started under, removes any leading directory path, and any trailing suffixes. A suffix is the part of a filename that follows a dot, including the dot itself. This is the service name used. For instance, if GNU Finger is installed as `/usr/local/bin/finger.new`, then the service '`finger`' is used. If it's installed as `/usr/local/bin/gfinger`, then the service '`gfinger`' is used. This behavior can be changed by the system administrator during installation.

3 Security Issues

One question that often arises when installing networking software which adds new functionality is whether it can be considered sufficiently secure. The most significant new function in GNU Finger with regard to security is the ability for a user to have a `.fingerrc` in the home directory. The following are the precautions take by GNU Finger:

- Check whether `.fingerrc` is writable to anyone except the owner. Notice that check is not enabled by default, since FSF users like anyone to be able to write any file – enable this check during installation by editing `config.h`.
- Check whether `.fingerrc` is owned by the user in whose home directory it's found. This, like the previous check, is disabled by default. It really only makes sense on systems where ordinary users can't give away their files.
- Execute the script through the user's login shell, using the command "*shell* -c *script*". This means that a user who has had his account disabled (i.e. shell set to a program that prints a notice or just dies) can't run a `.fingerrc` script. This behavior can be changed by hard-coding the shell in `config.h`.

4 Bug Reports

You are strongly encouraged to submit a bug report to bug-gnu-utils@gnu.ai.mit.edu if you have problems with this beta release of GNU Finger. Here are some things that are generally helpful to mention, when relevant:

- The GNU Finger version and where you obtained the distribution.
- Your hardware.
- Operating system.
- C Compiler used.
- The arguments you gave to `./configure`.
- Changes you made to the GNU Finger configuration files.
- Any changes you have made to the source code or installation procedures.
- Your network configuration; don't forget to mention what hardware, operating systems, and C compilers are used on each node where the bug manifests itself.
- Any problems you had during installation.

If you're having problems compiling or installing GNU Finger, then the following is particularly helpful to mention:

- The command line used to invoke `configure`.
- The current directory.
- The command line used to invoke `make` and `make` version (particularly if you're using GNU Make).
- A capture of the compiler output.

Please feel free to include any patches, as well.

5 Installation

5.1 Basic Installation

Here are the steps that you will need to take in order to install GNU Finger.

1. Pick a machine which will be the local finger server for your network. Create a `clients` file, and install it in `/usr/local/etc/fingerdir` (or the EtcDir as specified in `config.h`). Put the names of all hosts that should report to the finger server in this file. Don't forget to include the finger server itself.
2. For each client (the designated server is also a client), do the following,
3. Change your working directory to be the top of the GNU Finger sources. For instance, if you have placed the source in `/src/gnu/finger` you would type '`cd /src/gnu/finger`'.
4. In the source directory, type '`make clean`' and '`./configure`' if this host is different from the previous one. If you're using `csh` on an old version of System V, you might need to type '`sh ./configure`' instead to prevent `csh` from trying to execute `configure` itself.

   The `configure` shell script attempts to guess correct values for various system-dependent variables used during compilation, and creates the Makefile(s) (one in each subdirectory of the source directory). In some packages it creates a C header file containing system-dependent definitions. It also creates a file `config.status` that you can run in the future to recreate the current configuration.

   Running `configure` takes a minute or two. While it is running, it prints some messages that tell what it is doing. If you don't want to see the messages, run `configure` with its standard output redirected to `/dev/null`; for example, '`./configure >/dev/null`'.

   To compile the package in a different directory from the one containing the source code, you must use a version of make that supports the VPATH variable, such as GNU make. '`cd`' to the directory where you want the object files and executables to go and run `configure`. `configure` automatically checks for the source code in the directory that `configure` is in and in `..`. If for some reason `configure` is not in the source code directory that you are configuring, then it will report that it can't find the source code. In that case, run `configure` with the option '`--srcdir=DIR`', where DIR is the directory that contains the source code.

   You can tell '`configure`' to figure out the configuration for your system, and record it in `config.status`, without actually configuring the package (creating `Makefile` and perhaps a configuration header file). To do this, give `configure` the '`--no-create`' option. Later, you can run `./config.status` to actually configure the package for a particular host. This option is useful mainly in `Makefile` rules for updating `config.status` and `Makefile`. You can also give `config.status` the '`--recheck`' option, which makes it re-run `configure` with the same arguments you used before. This is useful if you change `configure`.

   '`configure`' ignores any other arguments that you give it.

   If you want to install the GNU Finger configuration files somewhere other than `/usr/local/etc/fingerdir`, then you should edit the files `./config.h` and `include/fingerpaths.h` now. You need to specify the alternate locations of where the configuration files will be kept. If you want to include the unsupported code for mugshots, then you should now also choose one of the face formats, as well as edit `lib/Makefile.in`, `lib/site/Makefile.in`, and `src/Makefile.in` to compile and link in the files necessary.

If your system requires unusual options for compilation or linking that `configure` doesn't know about, you can give `configure` initial values for some variables by setting them in the environment. In Bourne-compatible shells, you can do that on the command line like this:

```
CC='gcc -traditional' DEFS=-D_POSIX_SOURCE ./configure
```

The '`make`' variables that you might want to override with environment variables when running `configure` are:

(For these variables, any value given in the environment overrides the value that 'configure' would choose:)

`CC` C compiler program. Default is '`cc`', or '`gcc`' if '`gcc`' is in your search path.

`INSTALL` Program to use to install files. Default is '`install`' if you have it, '`cp`' otherwise.

(For these variables, any value given in the environment is added to the value that 'configure' chooses:)

`DEFS` Configuration options, in the form '`-Dfoo -Dbar ...`'

`LIBS` Libraries to link with, in the form '`-lfoo -lbar ...`'

5. To build and/or install the GNU Finger executables and standard targets, issue one of the following commands:

   '`make server`'
   To build and install all executables, plus install the finger-specific server configuration files. Use this if the host is the designated finger server.

   '`make client`'
   To build all executables, but install only those used by the non-server clients. Use this unless the host is the designated finger server.

   '`make all`' To build all executables, but perform no installation.

   The above commands build `lib/libfinger.a` and the main programs in `src`: `finger`, `in.fingerd`, and `in.cfingerd`, as well as `fingerd` on the server. If you want to, you can override the '`make`' variables `CFLAGS` and `LDFLAGS` like this:

   ```
   make CFLAGS=-O2 LDFLAGS=-s
   ```

6. Modify the system configuration so that the client has (refer to the system documentation for details on how to do this on a particular system):

   - Entries in the system `services` file, or equivalent, which mentions the correct TCP port for `in.cfingerd` (port 2003) and `in.fingerd` (port 79). If port 2003 is already used by something else, then read the section on Configuration Files for details on how to specify a port other than 2003 in the `/usr/local/etc/fingerdir/clients` file. A good name for the service is '`cfinger`'.

- Entries in the system `inetd.conf` file, or equivalent, which contains references to `in.cfingerd` and `in.fingerd`. `in.fingerd` needs to be run with UID `root`. Consult your system documentation for details on how to do this. `in.cfingerd` should be run with UID `root` on System V derivatives.
- a `/usr/local/etc/fingerdir/serverhost` file which contains the name of the GNU Finger server host,
- a `/usr/local/etc/fingerdir/mailhost` file which contains the name of the mail server to ask for user mail forwarding information and mailing list expansion, and
- the inetd daemon restarted, so that server and user requests can be answered.

7. When you have performed the above steps for each client, log onto the designated GNU Finger server.
8. Start the server daemon, `fingerd`. You should arrange to have `fingerd` started every time the server host is rebooted. For exact details on how to do this, please refer to the server host's system documentation.
9. If you chose to include the mugshots option, now might be a good time to install the mugshots. But first, try getting a face from another site running GNU Finger! For example, you might try

   ```
   finger --face bfox@aurel.cns.caltech.edu
   ```
10. Modify the files `x-.help` and `x-.site` in `/usr/local/etc/fingerdir/targets` for your site.

Now you're all set! You might like to read through the section on Configuration Files.

5.2 Configuration Files

This section describes the format of the GNU Finger configuration files.

5.2.1 The `clients` file

The `/usr/local/etc/fingerdir/clients` file contains a list of clients that the GNU Finger server `fingerd` is supposed to poll. You can edit this file and then send the finger server a `SIGHUP` to tell it that the configuration has changed. Each line in the file should be either the name of a host or a comment. The name can be preceded by `@`*`port`*, to tell the finger server to poll the particular host by using a port other than 2003. A comment is any line that starts with a hash sign (#). Below is a sample `clients` file:

```
# This file contains all GNU Finger clients on the gnu.ai.mit.edu
# network. Apple-gunkies is the GNU Finger server (see ``serverhost'').
apple-gunkies.gnu.ai.mit.edu

# Albert is the mail exchanger (see ``mailhost'').
albert.gnu.ai.mit.edu

# Spiff is a Sony, so port 2003 is already used for `mbanks'.
# Use port 2010 instead.
@2010 spiff.gnu.ai.mit.edu

churchy.gnu.ai.mit.edu
```

```
mole.gnu.ai.mit.edu
geech.gnu.ai.mit.edu
wookumz.gnu.ai.mit.edu
nutrimat.gnu.ai.mit.edu
kropotkin.gnu.ai.mit.edu
goldman.gnu.ai.mit.edu
hal.gnu.ai.mit.edu
wombat.gnu.ai.mit.edu
```

Although this sample `clients` file contains the fully qualified domain names of the hosts, it's usually enough to specify only the host name portion. Explicit IP addresses can be used too, but this is a practise strongly discouraged. Notice that the server is also in the clients file and has a `in.cfingerd`; this is necessary in order for the server to correctly poll itself.

5.2.2 The `serverhost` file

The `/usr/local/etc/fingerdir/serverhost` file holds the name of the GNU Finger server host; this is as the name implies, the host that the GNU Finger server `fingerd` runs on. Lines starting with a hash sign (#) are treated as comments. Below is a sample `serverhost` file:

```
# A-g does all the finger stuff
apple-gunkies.gnu.ai.mit.edu
```

5.2.3 The `mailhost` file

The `/usr/local/etc/fingerdir/mailhost` file holds the name of the mail exchanger host for the network. This host should know how to talk SMTP; this file should never hold the name of a host that can't. It's contacted to obtain mail forwarding information and to expand mailing lists if a `.forward` file can't be found in the user's home directory. GNU Finger always looks and reports on user `.forward` files regardless of whether `mailhost` exists or not. Any lines in this file that start with a hash sign (#) are treated as comments.

5.2.4 The `forwardhost` file

The `/usr/local/etc/fingerdir/forwardhost` file holds the name of the host to forward finger requests to when the current finger server can't find a matching user name or mail alias. No forwarding takes place if this file doesn't exist. Any lines that start with a hash sign (#) are treated as comments. This is a sample output of what it can look like when a request is forwarded:

```
% finger -l nosuchuser@gnu.ai.mit.edu
[No user nosuchuser@apple-gunkies.gnu.ai.mit.edu, forwarding
 request to life.ai.mit.edu]
Login name: nosuchuser In real life: ???
```

5.2.5 The `ttylocs` file

The `/usr/local/etc/fingerdir/ttylocs` file holds explanations for the hosts or terminal lines that users have logged in from or through. Each client host has its own copy of this file. Each line consists of a host name followed by a description. The name and description are separated by one or more blanks or TABs.

```
spiff.gnu.ai.mit.edu            NE43 Hall
```

```
susie.gnu.ai.mit.edu           NE43 Sony 427 x8568
spike.gnu.ai.mit.edu           NE43 Sony 427 x8568
apple-gunkies.gnu.ai.mit.edu   NE43 427
sugar-bombs.gnu.ai.mit.edu     Elsewhere
pogo.gnu.ai.mit.edu            NE43 447
albert.gnu.ai.mit.edu          Noisy Machine Room
128.52.46.42                   The salt mines
churchy.gnu.ai.mit.edu         NE43 426
mole.gnu.ai.mit.edu            NE43 430
geech.gnu.ai.mit.edu           NE43 426
wookumz.gnu.ai.mit.edu         NE43 427
calvin.gnu.ai.mit.edu          NE43
gnu.gnu.ai.mit.edu             NE43
kropotkin.gnu.ai.mit.edu       Total anarchy
```

5.3 Site Specific Functions

If you are interested in customizing GNU Finger's output, then the `lib/site` directory is the right place to start. If you would like to add new code for displaying faces, or have a particularly interesting "long" information output format, I would be glad to include it as unsupported code in the next release of GNU Finger. (I will direct correspondence regarding your code to you.) You can find other contributed code in `lib/site`, most notably different user info formats and code to handle different bit map file formats. The only file supported in `lib/site` is `userinfo.c`.

5.4 Configuration Options

The following definitions in `config.h` control the behavior of GNU Finger:

`SUPPORT_FINGERRC`
: Undefine to prevent users from writing `.fingerrc` scripts. Defined by default.

`CHECK_OWNER_FINGERRC`
: Define to make sure `.fingerrc` is owned by the user in whose directory it's found.

`CHECK_RDONLY_FINGERRC`
: Define to make sure `.fingerrc` isn't writable by anyone other than its owner.

`FINGERRC_SHELL`
: The shell to use to execute `.fingerrc`. Undefine to use user's login shell.

`DEFAULT_POLL_INTERVAL`
: Define to be the delay between polls, in seconds, unless an interval is explicitly given to `fingerd` with the '`--interval`' option.

`BASENAME_IS_SERVICE`
: Undefine if you always want the finger client to use the '`finger`' service. Otherwise the service is deduced from the basename of the client. Defined by default.

`INFO_IS_DEFAULT`
: Define if you prefer '`finger --info`' to be the default, undefine if you prefer '`finger --brief`' to be the default.

6 GNU Finger Quick Reference

```
finger options user@host
```

Finger obtains information about *user* on the remote system *host*. It can also be used to obtain information about mailing lists and special site-specific services.

Options can be one or more of:

‘`--info`’
‘`-i`’
‘`-l`’ Return as much information as possible about *user*.

‘`--brief`’
‘`-b`’
‘`-s`’ Only display current login info.

‘`--face`’
‘`-f`’ Display a mugshot of *user*. Not supported.

‘`--port` *port#*’
‘`-P` *port#*’ Connect to the finger server using port or service *port#*.