mgetty+sendfax

Version 1.1.28

January 2002

Gert Doering
gert@greenie.muc.de
gert@space.net

1 Introduction

mgetty allows you to make optimum use of your modem or fax modem in a unix environment. mgetty handles incoming calls without interfering with outgoing calls. If the modem can do fax class 2 or 2.0, mgetty can also receive faxes.

sendfax is a standalone backend program to send fax files.

This manual explains how to configure and install the package on various operating systems and modems, and how to use those programs.

1.1 Copying conditions and (lack of) warranty

WARNING: This package is still BETA software. Use it at your own risk, there is **no** warranty. If it erases all the data on your hard disk, damages your hardware, or kills your dog, that is entirely your problem. Anyway, the program works for me and quite a lot of other people.

If you want to redistribute mgetty+sendfax under a different license (like in "selling it to your customers"), contact me, and we will work something out.

1.2 Features of mgetty and sendfax

This package contains two major programs, mgetty and sendfax.

This is what you can do with sendfax if you have a standard class 2 fax modem:

- send faxes directly or using shell scripts
- do "fax polling", this means you can call the weather station and get them to send you a fax containing the current weather map. (Not all modem manufacturers implement this feature in their modems!)
- create a "fax queue", outgoing faxes get sent automatically, the user is informed by mail about the result.

mgetty allows you to use your modem line for receiving fax and data calls, without hindering dial-out on the same line.

- mgetty knows about "smart" modems, and will make sure that the modem is always in a defined state (specific modem initialization possible)
- Incoming calls are answered manually (RING -> ATA -> CONNECT) instead of using auto-answer ('ATS0=1'), this way the modem won't pick up the phone when the machine is down or logins are not allowed.
- mgetty incorporates all features of uugetty: it honours 'LCK.*' files created by uucico and other comm programs, this way it will prevent dial-outs while a caller is online and it won't be confused if other programs use the modem.

- `mgetty` can receive faxes (if your modem supports fax class 2 or 2.0).
- `mgetty` knows about incoming FidoNet calls.
- `mgetty` has extensive logging / debugging features
- do “fax poll sending”, that is, you can setup your machine as fax poll server, to send some fax pages to “fax poll” callers. (Send informations about your system, the current wheather map, ...). Be warned, even less modems support this feature.
- `mgetty` can selectively refuse calls based upon CallerID, if your local Telco supports it and your modem knows how to decode and report it. CallerID is also logged.
- `mgetty` has facilities to allow you to refuse incoming FAXes when available disk space is low.

If you have any bug reports, suggestions, please report them to gert@greenie.muc.de (or, if [and only if!] that doesn't work, to gert@space.net).

Also, I have created a mgetty mailing list, for discussion of problems and suggestions. You can subscribe by sending a request to mgetty-request@muc.de (forwarded to Crynwr.com for precessing) and you can send articles to the list by sending them to mgetty@muc.de. The list manager will write you back for verification of your address, so make sure it is correct.

The mailing list is currently gated bidirectionally into the newsgroup de.alt.comm.mgetty. In spite of being in the German language hierarchy, the language in the group is English. Posts in German should be ignored.

The mailing list is archived on a WWW site, look at ‘`http://www.elilabs.com/mgarc/index.html`’ (many thanks to Robert J. Brown, rj@eli.elilabs.com). It's also archived by Marc Schaefer, on ‘`http://search.alphanet.ch/`’ **(NEW!)**. The latter search engine indexes somewhat more than only the mgetty list, so you might want to add 'mgetty' to your query to restrict it.

1.3 Supported systems and modems

Mgetty has been successfully installed and run on the following systems (though older systems might need work to get current mgetty versions to compile):

SCO Unix 3.2.1 (ODT 1.0)	(well tested)
SCO Unix 3.2.4 (ODT 2.0 + 3.0)	(very well tested)
SCO Open Server 5.0	(well tested)
Linux (everything from 0.99pl1 up)	(very well tested)
ISC Unix 3.0	(tested)
SVR4 Unix	(well tested)
SVR4.2 Unix	(needs more testing)
AT&T 3B1 3.51m	(tested)
HP-UX 8.x and 9.x	(well tested)
AIX 3.2.5, 4.1, 4.2 and 4.3	(very well tested)
SunOS 4	(well tested)
Solaris 2.x	(well tested)
NetBSD / FreeBSD (all versions)	(very well tested)
Apple Darwin (Kernel 5.1)	(tested)

It should be possible to run mgetty on any other Unix with '`termio.h`' or '`termios.h`'. For best results, use of the library functions `select(S)` or `poll(S)` is recommended, but there's a workaround if your system hasn't either. (Warning: for Unix SVR3.1 or earlier, *do not use poll()*, it will not work on tty devices.)

Up to now, it has been successfully used with the following modems (no exhaustive list) in fax mode:

ZyXEL U1496 (various ROM releases)
(very well tested, a couple of problems remain, depending on the ROM release)

ZyXEL 2864/2864I (various ROM releases)
(very well tested, some firmware versions have problems)

USR Courier/Sportster series
(well tested, Couriers work great, Sportsters are ok for data)

MultiTech (various models)
(tested, works very well, very good fax implementation)

SupraFAX v32bis
(tested, works well, no fax polling available)

GVC FM144/+
(tested, works well, no fax polling available)

TKR DM-24VF+ (Deltafax)
(tested, works quite well)

Zoom V.FAST 24K/28K
(tested, works, some problems with fax/data distinction)

ELSA MicroLink (various)
(tested, usually works well, some firmware versions have issues)

It *should* work with all class 2 faxmodems. Maybe the DC2 character sent at the beginning of a page by '`faxrec.c`' must be changed to XON, for very old class 2 modems (implementing very old drafts of the standard). See Section 5.1 [Modems], page 31.

In Data mode, it will work with every Hayes-compatible modem.

1.4 Configuration and installation

Compiling of the package should be quite straightforward. You have to copy '`policy.h-dist`' to '`policy.h`' and edit it to set some local policy options, see the comments in that file. (Most default values should be fine, though).

Then, edit the '`Makefile`', to specify installation paths, some system defines and some system dependent libraries (explained there).

After that, a 'make' should build the programs and the documentation.

Before you can run 'make install' to setup and install everything, make sure that the user that you specified for 'FAX_OUT_USER' in 'Makefile' exists in your system. This user will be needed to run faxrunq or faxrunqd, and owns all the outgoing fax queue (this has been changed for 1.1.29 - earlier versions had the fax queue world writeable, which was a security problem).

If your compiler complains about the #ident lines I use for 'RCS', please run make noident, that will take care of those lines (this is only needed if the compiler refuses to go on - some compilers complain, but go on anyways).

If you get an error message about "unresolved symbols" when linking, you may have to tell the package whether you have the select(S) or poll(S) system calls, by defining -DUSE_SELECT or -DUSE_POLL flags in the 'Makefile' (If you don't know it, try both, until the error goes away). If it's not related to select or poll, please check the systems man pages which libraries to link, and add appropriate -l<library> statements to LIBS.

If your system has neither the select(S) call nor the poll(S) call, mgetty is not fully operational—these functions are the only way to prevent mgetty from eating up characters when some other process is trying to dial out.

You can use mgetty anyway, by specifying -DUSE_READ in the Makefile, but beware: with this, mgetty will eat up at least one character of the modem response when another program is dialing out. That may lead to disastrous results if e.g. the 'CONNECT' string is thus corrupted, but most of the time, the character that 'mgetty' eats away will be an cr or nl or part of the command that the modem echoes back.

If you have any problems compiling mgetty and sendfax (on a Unix-like system—I do not support MS-DOS or Windows!), please contact me. *But make sure that you have read the documentation!*

Later on in this document you can find some more detailed instructions about specific operating systems (Linux, FreeBSD, and so on).

1.5 Runtime configuration: Overview

If mgetty or sendfax are run "as is", they will use their compiled-in defaults from 'policy.h'.

If the configuration files 'mgetty.config' (see Section 2.9 [runtime-mgetty], page 10) and 'sendfax.config' (see Section 3.8 [runtime-sendfax], page 25) exist (and are readable), both programs will get their run-time configuration from there. Items not specified there will still be taken from the compiled-in defaults. Command line switches will always override those settings. The configuration files are usually located in '/etc/mgetty+sendfax/'.

If you specify command line arguments (see the mgetty(1) and sendfax(8) man pages for details), this will override both compiled-in and config file defaults.

2 Using mgetty

You can't simply call mgetty from a shell script (like '/etc/rc') or interactively, because login is not possible unless mgetty is called directly by the init process. The next sections explain how to do this.

2.1 How mgetty works

To help you understand how mgetty works, here is an example of what happens in various circumstances when you use it to control a modem connected to a serial line, e.g. '/dev/tty2a'.

When the computer is booted, the operating system starts the init process, which is responsible for making sure that gettys are running on the appropiate i/o devices, e.g. virtual terminals, serial lines and modems. init reads its configuration file, '/etc/inittab' (on System V), which tells it that the line '/dev/tty2a' should be controlled by mgetty. It then creates an entry in '/etc/utmp' (login needs this, that's why you can't log in if you try to start mgetty by hand), and forks a new mgetty process, using the command line specified.

When mgetty is started, it first checks if a valid lock file held by another process exists. If it does, this means that the port is in use, and mgetty will wait until the lock file goes away. Invalid lock files, e.g. for nonexistent processes ("stale" locks), are ignored.

Once the port is free, mgetty creates its own lockfile, initializes the modem and removes its lock file again. Then it waits for something to happen on the port. Note that it does not *read* any characters, it just checks if there are any available for reading by using poll() or select().

There are two possibilities once characters arrive, either a different program (e.g. uucico) has started dialing out or a 'RING' was sent by the modem. In the first case, mgetty should leave the port alone. This is easy *if* the program dialing out has created a valid lock file: mgetty will find it, wait for it to go away and then exit (which will cause init to start a fresh mgetty process, which will then wait for the next call).

In the second case, when there is no lock file, mgetty assumes that the phone is ringing, creates a lock file and reads the characters available. If it finds a 'RING', it picks up the phone by sending 'ATA' and waits for the 'CONNECT' message. If the caller is a fax machine, it saves the fax in the directory 'FAX_SPOOL_IN' (usually '/var/spool/fax/incoming') and exits. If it is a modem, it prints '/etc/issue' and displays a login prompt. Once it has received a login string, it calls /bin/login and lets it handle things from here. login will read the password and will then start the user's login shell, uucico, a dialup SLIP link or whatever, but mgetty doesn't care about that. The lock file remains so that no other programs will try to use the modem while somebody is logged in.

(If the 'login.config' configuration file is used, mgetty can also call other login programs than /bin/login. See below for more details)

Once mgetty has terminated for whatever reason, init might reinitialize the port (this is why mgetty waits for lock files to go away instead of quitting immediately) and will then start a new mgetty process, which will remove lock files left over from the last login.

The lock file handling is tricky, but very important. It is essential that *all* programs that use the modem agree on one locking protocol, otherwise one program might not know that the modem is in use and will try to dial out anyway. A typical lock file is a file called '/var/lock/LCK..ttyxx', containing the process ID (PID) of the process currently using the modem. Other processes can read it and tell if the lock file belongs to an existing process or if it is "stale" and can be removed. This will obviously not work if the processes look for lock files in different places, or if one of them writes its PID in ASCII and another one tries to read it as a binary number (while mgetty and sendfax do not care whether foreign lock files are written in binary or ascii format, other programs do! mgetty can sometimes detect this problem, and will then log a warning).

2.2 The '/etc/inittab' entry

An typical 'inittab' entry for mgetty looks like this (on SystemV-style OSes):

```
<tt>:rlevel:<respawn|off>:/usr/sbin/mgetty [options] <device>
```

where 'tt' is a short form of the 'device' name, used by init and who for internal purposes. Usually this is something like 'S0' or '2A' or so.

'rlevel' specifies the runlevel that the command in the fourth field is run at, this may be '23' or '56' or so, look at man init and the existing '/etc/inittab' on your system.

The next field tells init whether that entry is active (respawn) or not (off), and the fourth field specifies the full path of the program to run.

The following options are available for mgetty:

'-x <*level*>' sets the debugging level. This is very important for diagnosing problems, as with higher levels, mgetty will write very detailed informations about its internal workings to its log file.

'-s <*speed*>' sets the port speed. If not specified, the default from 'policy.h', (definition DEFAULT_PORTSPEED) will be used.

'-k <*space*>' sets the minimum number of kbytes required on the incoming FAX spool directory. If there isn't this much space in the spool directory, the connection is terminated. The default is 1 megabyte.

'-m *'expect send ...'*' sets the modem initialization sequence.

'-r' for direct lines (no modem chats are done)

'-p <*login prompt*>' sets the login prompt (various escapes are allowed)

'-n <*rings*>' sets the number of RING messages to expect before sending ATA to answer the phone. Default is one RING.

'-R <*sec*>' tells mgetty to enable "ring-back" or "ring-twice" mode. This means that mgetty won't pick up a call immediately, but the caller has to hang up after the first ring, and call again in the next <*sec*> seconds.

'-i '/etc/issue'' specifies the issue file to display before prompting for login.

'-S '<fax_document>'' specifies the document(s) to send to polling fax machines (full path required). '<fax_document>' has to be in G3 format (as for sendfax), or a text file listing G3 fax files (one file per line).

A sample entry in '/etc/inittab' might look like this:

```
F1a:23:respawn:/usr/sbin/mgetty -x 3 tty2a
```

For a more detailed explanation of all the options, please look into the '`mgetty(1)`' man page.

2.3 Choosing the right device

Some operating systems provide two separate devices for each serial line, one of which is intended especially for gettys. This is *NOT* a good idea, because it won't work with `mgetty`. It is meant for modems used in auto-answer mode.

`mgetty` picks up the phone by hand, this will only work properly if you use the *same* device you would use for dialing out. (Some people like to create a symlink '`/dev/modem`' for this, but you have to be consistent and use this for *all* comm programs if you do - otherwise you'll run into problems with the lock file names).

Here are some examples for different systems:

- SV Unix systems using the FAS serial driver:

  Use `ttyF01` etc., (minor number 80+(port number). Do *not* use `ttyFMxxx`—mgetty will open the device anyway, but then an open to the "normal" port (without carrier detect) will block.
- SCO Unix systems with the stock serial driver:

  Use the modem-control port (`tty2A`), *not* `tty2a`, because hardware flow control and carrier detection do not work on "lowercase" tty devices.

  The same holds for "smart" serial boards, as specialix, digiboard, ..., because they mimic the behaviour of SCO's sio driver.
- Linux:

  Use '`/dev/ttyS*`', *not* '`/dev/cua*`'. See Section 5.2.4 [Linux], page 47.
- SunOS, FreeBSD, NetBSD:

  Use '`/dev/cua*`', not '`/dev/ttyS*`'. Don't ask me why it has to be this way, but the other way won't work. (On SunOS or Solaris, you can find some gory details in the `man zs` man page).

2.4 Log files

`mgetty` will normally write all actions into a logfile, named '`LOG_PATH.`<*device*>' (LOG_PATH is defined '`policy.h`'), e.g. '`/var/log/mgetty.log.ttyxx`', so you can easily see what's happening in your system. (If a fatal error occurs, the error message is also written to the console, and if that is not possible, mailed to '`ADMIN`'). The name of the log file can be changed in '`policy.h`'.

If `mgetty` is compiled with `-DSYSLOG`, auditing and error log messages will also go to `syslog` (if your system supports it).

2.5 Denying logins

If you want to prevent mgetty from accepting calls, you can do so by creating a file called '/etc/nologin.<device>' (e.g. '/etc/nologin.tty2a'). If mgetty detects an incoming call, and sees this file, it will *NOT* answer the phone. This way the caller does not lose money for a useless call when logins are not allowed anyway. (You can change the filename that is checked by changing `NOLOGIN_FILE` in 'policy.h')

This can be used for quite sophisticated scheduling purposes - imagine a modem attached to a line used for voice during the daytime. So, you want the modem only to answer the phone during 9 pm and 7 am. What you do is to have `cron` create a '/etc/nologin.*device*' file at 7 am (so the modem won't answer the call during the day) and remove it at 9 pm (so the modem can answer the phone at night).

Naturally, this can be easily extended - e.g., answer the phone only on weekends (similar cron job), don't answer the phone if there are less than 5 Mbyte free on disk (have a process check for free disk space regularly and create '/etc/nologin.*' file(s) accordingly), ...

2.6 Direct serial lines

If you have a direct serial line between two unix machines, or a unix machine and something else, and want to run a getty program on the unix side that should not interfere with outgoing traffic (e.g.: two unix machines, both sides running gettys, both sides able to initiate an uucp connection), you can also use mgetty. Start it with the '-r' flag (as with uugetty), and it will not try to initialize the modem or wait for RINGs. It will just sit silently on the port and wait... You won't see '/etc/issue' on the other side until mgetty gets at least one character, this prevents two mgettys from talking to each other.

This may be valid for leased lines with modems in leased line mode as well, but that may depend on your modem setup.

2.7 Interaction between mgetty and other programs

Normally, after a caller enters his login name, `mgetty` calls `/bin/login` to do the password checking and system login.

In some special cases, you may want to call other programs instead of `/bin/login`. For example, you could want to call `/usr/lib/uucp/uucico -L <username>` for all login names starting with 'U*' (to have `uucico` do the authentication, works only with taylor uucp 1.05 or taylor uucp 1.04 with my patch in 'patches/taylor.p1'), or `/usr/lib/fnet/ifcico` for incoming FidoNet calls (using `ifcico` from Eugene Crosser's 'ifmail' package).

`mgetty` can do all this. It's controlled by a configuration file 'login.config', normally located in '/etc/mgetty+sendfax/' (definition `LOGIN_CFG_FILE` in 'policy.h'). I have provided a sample file with lots of comments, please look into that file for the syntax to use. To make `mgetty` understand incoming fido calls, you have to compile it with `-DFIDO`.

If you are worrying about security, you can also use this mechanism: just call `/bin/login` only for trusted, known users, and `/bin/false` for every other login name - so, only those listed in 'login.config' will be able to log in.

This mechanism can also be used to automatically start up a `PPP` server if an incoming client sends PPP packets. This feature is called *AutoPPP*. For it to work, you have to compile `mgetty` with `-DAUTO_PPP` (added to CFLAGS in Makefile). After this, `mgetty` will detect incoming PPP packets, and run the program that is specified in '`login.config`' by the special user name "/AutoPPP/". See the sample '`login.config`' file for an example.

Which options should be specified depends on your PPP server program and your local setup. Don't ask me about that – I wouldn't know. Instead, please check the relevant man pages.

2.8 Using Caller-ID to selectively accept or reject calls

Some telephone companies provide a service to the subscriber, called "Caller ID", where the phone number of the caller is transmitted while your phone is ringing. Not all providers support it, and you'll have to ask for it.

If your modem is able to retrieve callerid information, and `mgetty` is compiled with `CNDFILE` defined in '`policy.h`', `mgetty` can check the caller's number before answering the phone. (Right now, it works for most variants out there, ZyXEL, Rockwell, ELSA and isdn4linux among them). If `CNDFILE` is undefined, or if the file specified does not exist, all calls will be allowed.

One important thing: for most analog modems, you **must** set the number of RINGs to wait for to two (2) or higher (set '`rings 2`' in '`mgetty.config`'), because the ID code is sent between the first and the second RING. If mgetty picks up the phone too soon, the modem can't get this information.

Whether a call is accepted or denied is controlled by the configuration file set with `CNDFILE` in '`policy.h`'. The usual default is '`/etc/mgetty+sendfax/dialin.config`' (a sample file is installed per default).

That file contains a series of tokens separated by newlines, commas, tabs and spaces. The callerid number is compared with each token in turn, until a match occurs. A match occurs when the token compares equally to the callerid information up to the length of the token. If the token is prefixed with a "!", a match means "do not answer the phone". The token "all" matches any telephone number, and will terminate scanning of the cndfile. If no callerid number is present, it is assumed to have the value "none". A line starting with "#" is a comment. There is an implicit "all" at the end of the file.

For example:

```
# list of my friends' data lines
3433535, 7445343, 5551212
# dad's fax
4164646777
# disallow [other] calls from numbers matching the following prefix:
!416
# disallow that speed dialer that keeps hitting my machine
!3444444
# allow all calls with the following prefixes
832, 555
# don't allow calls when there's no callerid:
```

```
!none
# It's okay to accept calls from out of area
# ("OUT_OF_AREA" token seems ZyXEL specific)
OUT_OF_AREA
# disallow all other calls
!all
```

For the future, Chris Lewis is planning on adding special modem initialization strings (e.g., 2400 bps only, fax-only, ...) dependant on the caller number.

For most applications, this kind of static configuration is enough. If you have special needs, you can choose to run an external program to decide this. The program name is configured with the `cnd-program` statement in '`mgetty.config`'. Its command line arguments are:

```
<program> <tty> <CallerID> <Name> <dist-ring-nr.> <Called Nr.>
```

`CallerID` is the number of the caller, if known, or '`none`', if not. `Name` is the name of the caller, or empty ('') if unknown. `dist-ring-nr.` is the RING type, if you have "distinctive RING" on your telephone line and your modem supports this, or "0" for an unidentified call. `Called Nr.` is the number that was called (this is only meaningful if you have ISDN, and your modem signals the complete number called to the host - e.g. ELSA or ZyXEL 2864I do that).

For example, a call from 12345 to 56789, using ISDN, coming in on ttyS3, could lead to a program called like this:

```
check.cnd ttyS3 12345 '' 0 56789
```

The program return value decides whether the call should be accepted. Currently, the following values are defined:

0 - accept call, no specific preferences
1 - reject call

Future versions will allow external selection of the way mgetty/vgetty is supposed to answer the call (data/voice/fax/...), but that's not implemented yet.

Note: this can not only be used to decide whether to accept a call or not. You could as well use it to display the caller ID on an external LCD screen, in an X11 window, print it, initiate a D-Channel Callback, or do whatever you want that needs the Caller ID data.

Note2: be careful what kind of programs you call! They run with user id 0 (root), so that could easily be a security risk if you're not careful.

2.9 Runtime configuration for mgetty: '`mgetty.config`'

`Mgetty` works quite well with the compiled-in defaults (it has been the only way to configure it for a long time), but that's quite unflexible, and especially if you use different modem types, it's extremely unhandy. The built-in defaults can be modified by command line options, but that's not perfect either, because it makes '`/etc/inittab`' entries very long and difficult to read.

If compiled with *config file* support (define `MGETTY_CONFIG` in '`policy.h`'), `mgetty` can use a configuration file, quite similar to those which "Taylor UUCP" uses, which make dynamic setup far easier.

The config file is usually located in '/etc/mgetty+sendfax/' and named 'mgetty.config'. Its format is very simple. Each line contains one keyword, and possibly arguments for it, separated by whitespace. Empty lines, and comment lines (lines starting with '#') are allowed.

The config file is grouped into port-specific sections, separated by `port` *<tty-name>* lines. Everything before the first `port` line specifies global defaults, everything between two `port` statements specifies configuration items valid only for exactly this device. Let me show you an example:

```
# global defaults:
# fax station id is always the same
fax-id ++49-89-1234
# per port stuff
port tty1a
# This modem can't fax
modem-type data

port tty2a
# more verbose logging for this modem
debug 9
```

The data part of each line, following the keyword, can be a string (in most cases), a chat sequence (a series of strings, separated by whitespace, that specify the "modem talk" to do. It starts with "expect" string, then "send", then "expect" again, and so on), an integer (interpreted as decimal, octal or hexadecimal, depending on the leading character [1-9/0/0x]), or boolean ('y(es)' or 't(rue)' vs. 'n(o)' or 'f(alse)'). If no argument is specified, this will be considered "value not set" (if allowed) or "error" (if value is mandatory), except for boolean values. In that case, it's interpreted as 'true'.

Many of those configuration items can be overriden from the command line. In that case, command line options take precedence over configuration file settings (and those take precedence over built-in defaults). In many cases, the built-in defaults can be set in 'policy.h'.

The available configuration items are (command line options, if available, given in brackets):

– `speed` [-s] *port speed*

  Specify, as integer value, the port speed to use. Default is `DEFAULT_PORTSPEED`. If the given speed is not valid, `mgetty` complains loudly and exits.

– `switchbd` *fax recv. speed*

  Some modems, mainly Rockwell chipsets, switch to 19200 bps when entering fax mode. Others may need other speed switches (but I know none). If your modem is Rockwell based, try `switchbd 19200` if fax reception doesn't work. (**Warning:** if this is set wrongly, fax reception will definitely fail. For most sane modems, you do **not need** this.). Default is `FAX_RECV_SWITCHBD`.

– `direct` *yes/no* [-r]

  Tells mgetty that it is running on a direct line. Mgetty won't try to initialize any modem, nor will it wait for 'RING'. It will just wait for any character, and then output the issue file and login prompt. This option is used if you want to connect to machines via nullmodem cable. Default is *no*, since `mgetty` is designed for modems. . .

- `blocking` *yes/no* [-b]

  Tells mgetty to open the device in '`blocking`' mode, that is, the `open()` system call won't succeed until carrier detect is set. This is set if `mgetty` is called as `getty`. I'm not sure whether it's very useful, but I include it for completeness. Default is *no*.
- `port-owner` *username/userid*

  If set, mgetty will `chown` the tty line to the given username (you can specify a string or an integer uid, but the integer must be valid). This is highly recommended for security purposes: only give port access to those users you trust not to misuse your modem lines! Default is `PORT_OWNER`.
- `port-group` *groupname/gid*

  If set, mgetty will `chgrp` the tty line to this group id (which can be given as group name, or as integer gid). If it's not given, or not valid, the primary group id of '`port-owner`' will be used. Default is `PORT_GROUP`.
- `port-mode` *permissons*

  Specifies the permissions to `chmod` the device to. **Never** make a modem device world-accessible, better use '`0660`' or even '`0600`'. Default is `PORT_MODE`.
- `toggle-dtr` *yes/no*

  Tells mgetty whether it should lower the DTR line upon startup to reset modem. Default is '`yes`', but some (few) modems react allergic to that and crash.
- `toggle-dtr-waittime` *msecs*

  Specifies the time to hold the DTR line low. Default is 500 milliseconds.
- `data-only` *yes/no* [-D]

  Tells `mgetty` to forget about faxing and only use the data part of the modem. Default is '`false`'. You need this if your modem can't distinguish incoming fax and data calls.
- `fax-only` *yes/no* [-F]

  Tells `mgetty` to put the modem in fax-only mode. You need this if your modem can't distinguish incoming fax and data calls, but you need fax more important than data; and you need it if you want to disable data calls for security reasons (this could be achieved via '`login.config`' as well)

  Watch out: if you have setup some unusual `answer-chat`, incoming calls might still come through. So check your setup!
- `modem-type` [-C] *mtype*

  Specifies the kind of modem connected to the port. Default is `DEFAULT_MODEMTYPE`. Valid options are:

  - auto

    Mgetty will detect the modem type itself (which may occasionally be not desirable, or it may fail on old modem equipment). Mgetty will use the `ATI` command to find out the modem type, and select the proper fax class accordingly. If that fails (unknown modem type), mgetty will try class 2.0 and then class 2.
  - c2.0

    Modem is a *class 2.0* fax mode. Works better than class 2, if both are available, because its better standardized. Known to work with USR, ZyXEL 1496 and 2864 series, and ELSA modems.

- cls2

  Modem is a *class 2* fax modem, mgetty will not try class 2.0.
- c2.1

  Modem conforms to the new ITU T.32 standard (class 2.1). To my knowledge, there are no such modems available yet, but supporting them will be easy as class 2.1 is very similar to class 2.0.
- cls1

  Modem can only do class 1 fax. NOT IMPLEMENTED YET. (And not recommended anyway).
- c1.0

  Modem can do class 1 fax conforming to ITU T.31 standard. This isn't much better than class 1 (use class 2 / 2.0 if available!), but is better standardized. NOT IMPLEMENTED YET.
- cls2ok

  (obsolete, use `modem-quirks 02`)
- data

  Do not try fax initialization, same as if '`-D`' given.

There is no way (yet) to tell mgetty to use **only** fax mode and refuse data calls with this option, use the `fax-only true` statement for that.

– `modem-quirks` *bitmask*

Some modems have a very peculiar interpretation of the fax standards. Some of the internal operations of `mgetty+sendfax` can be adapted to that. The argument is a number, constructed from valies in '`fax_lib.h`', one bit per "quirk". Usually you won't need this option, because those modems really needing it are auto-detected and handled properly anyway.

Right now, the following quirks are defined:

0x01 leave the modem in class 2 mode instead of switching
to class 0 before sending ATA (you might try this if
adaptive fax/data answer doesn't work).
0x02 class 2 bit order is correct (MultiTech) - unimplemented
0x04 do not trust +FPTS:x,lc,blc values
0x08 do not wait for XON character when sending pages
0x20 AT+FCC/+FMINSP bug workaround for (very) old USR Courier V.32
0x40 display incoming informations about 'non standard frames' - this
might be necessary on some USR modems to work around logic bugs

– `init-chat` [-m] *expect send expect send ...*

Tells mgetty the chat sequence to use for initializing the modem. **Warning**: the sequence starts with **expect**, which will in most cases be '`""`' (nothing). This ordering was chosen because UUCP does it this way, and I wanted to avoid confusion here.

Example:

```
init-chat "" ATQ0E1V1H0 OK ATL0M0S0=0 OK AT&K3 OK
```

– `force-init-chat` *expect send expect send ...*

  In some cases, the modem can get stuck in a mode where it won't react to a simple `AT` command. Usually this happens because the modem is set to ignore a DTR drop and still has a data connection to the other side. If you use a voice modem, it could be stuck in voice mode.

  In these situations, the normal `init-chat` will time out, because the modem won't send the proper responses back.

  To get the modem back into a sane state, you can use the `force-init-chat` chat sequence. The default setup will send the DLE ETX characters, to get voice modems back to life, and then the `(pause)+++(pause)ATH0` sequence to get the modem back from data mode to command mode.

  You could prepend this sequence to `init-chat` (it wouldn't harm), but especially the pauses around the `+++` sequence makes this undesirable slow.

– `post-init-chat` *expect send expect send ...*

  Some modems forget parts of their settings when going from data to fax mode and back during modem initialization. For example, some USR models forget the settings of "Caller ID delivery" (AT#CID=1), and some ELSA modems forget their current DTE port speed when going from voice to data mode, thus leading to RING messages being delivered with the wrong baud rate.

  For those modems, you can use this command to set up some AT commands that are executed after all other fax and voice initialization has been done. Be careful with what you do! If you send an ATZ (modem reset) or something similar here, all your fax/voice settings will be lost!

– `modem-check-time` *seconds*

  Some modems have the nasty tendency to crash silently. With this option, you tell `mgetty` to check every *seconds* seconds with a simple '`AT...OK`' sequence whether the modem still reacts. If not, `mgetty` will restart itself and do a full modem reset. Default is `MODEM_CHECK_TIME`

– `rings` [-n] *nnn*

  Sets the number of `RING` messages to wait for, before mgetty picks up the phone. Default is 1. **Warning:** if your modem auto-answers, for whatever reason, set this to something **different** than the value set with `ATS0=mmm`, otherwise the modems autoanswer and mgettys manual answer will collide (most modems hang up if a command is received during auto-answer)

– `msn-list` *msn1 msn2 msn3...*

  If you have an ISDN modem that signals the called party number (MSN) to the host, you can use this statement to map the MSN numbers to distictive RINGs. The MSN called will be compared the list, and the first match is used for the distinctive RING number. The list is searched from left to right.

  This is known to work with ELSA and ZyXEL ISDN terminal adaptors.

– `get-cnd-chat` *chat sequence*

  This is needed if you have a modem that supports "caller ID" detection, but needs a special command to get the CID information. Right now, this is only needed for some ELINK ISDN adaptors (see Section 5.1.18 [Elink-ISDN], page 43), most other CID-capable modems send the CID on their own and don't need this.

Don't forget to set `rings` to at least 2, otherwise the CID grabbing code won't work.

– `cnd-program` *pathname*

  Specify a program to be run before answering an incoming call. Use this if the static Caller ID selection in CNDFILE (policy.h) is not sufficient, or if you want to use the Caller ID data for other purposes (displaying, for example). See Section 2.8 [Caller-ID], page 9.

– `answer-chat` *chat sequence*

  This is the command sequence that is used to answer a phone call. Usually you can leave it at the default ‘ `"" ATA CONNECT \c \r` ’, but for some modems you need ‘`ATS0=1`’ in place of ‘`ATA`’ (ATA not allowed). The extra ‘`\r`’ expect string is needed that the code can grab the full *CONNECT xyz\r* string. It will work without the \r, but then the logging information will be less detailed. **Right now, \r won't work at all, it's not implemented yet. Don't use it.**

– `answer-chat-timeout` *secs*

  During the *answer-chat*, each "expect" string must be seen in the time specified here. Default is 80 seconds. This time should be at least some 5 seconds longer than the time set with the `ATS7=...` modem setup command.

– `autobauding` *yes/no* [-a]

  Some modems switch their DTE line speed to the communication line speed after connecting, e.g., after sending ‘`CONNECT 2400`’, the modem switches to 2400 bps. Newer modems usually have a switch to "lock" a DTE baud rate, which is strongly recommended. If your modem insists on doing this speed switch, setting `autobauding` to *true* will make mgetty behave accordingly.

– `ringback` *yes/no* [-R]

  If you have to put your modem and your telephone on the same phone line, you can switch on "ringback" or "ring-twice". This means, mgetty won't answer the phone on the first call, but remember the call, and pick up on the second call (if it comes in the time specified by `ringback-time`).

– `ringback-time` *secs*

  This setting specifies how much time may pass between the first and the second call if "ringback" is active. Default is 30 seconds.

– `ignore-carrier`

  If your Modem does not assert the DCD (carrier detect) line, or the serial port or cable or serial driver is broken, it is possible that `mgetty` or `login` will block after a successful CONNECT (that means: everything seems to work, but suddenly nothing is sent to the port anymore. Depending on the operating system used, this can be before printing the ‘`/etc/issue`’ file or not before printing the ‘`password:`’ prompt.

  To work around this, you can switch off the carrier detection in the kernel: set `ignore-carrier true`. Default is `false`.

  WARNING: If you use this, your system won't be able to detect when a caller just hangs up instead of cleanly logging out. This may result in hanging modems, etc.

– `issue-file` [-i] *file*

  This is the file printed before the login prompt. Default is '`/etc/issue`'. Some special characters are substituted by connect speed, date, etc. - see below (login-prompt) for a list.

– `prompt-waittime` *msecs*

  This specifies how long `mgetty` will wait for modem and line to settle down before printing issue file and login prompt. Default is 500 milliseconds.

– `login-prompt` [-p] *prompt*

  This specifies the login prompt that mgetty will output. Some special characters in this string (and in the issue file, btw) are recognized and replaced by something else:

  - `@` system name
  - \n newline
  - \r carriage return
  - \g bell
  - \b backspace (ascii 010)
  - \f form feed (ascii 013)
  - \t TAB
  - \P (and \L) port name (e.g. ttyS0)
  - \C date and time, in "ctime()" format
  - \I Connection string (e.g. 2400/REL)
  - \N (and \U) number of users currently logged in
  - \S Port speed (e.g. 38400)
  - \D current date in dd/mm/yy format
  - \T current time in hh:mm:ss format
  - \Y CallerID of the current caller
  - *digit* character with the specified octal code

  The maximum lenght of the login prompt is limited to 140 characters (after expansion).

– `login-time` *secs*

  This specifies the maximum time the user can take to log in. If no login has occured after that time, `mgetty` will hang up. Default is `MAX_LOGIN_TIME` from '`policy.h`'.

– `fido-send-emsi` *yes/no*

  Only relevant when mgetty was compiled with `-DFIDO`. Controls whether mgetty should send a FidoNET style "EMSI_REQA77E" packet before prompting for login. Default is on. Switch this off if you have FIDO support compiled in but experience weird problems with some PPP clients (or users!) being confused by that string.

– `login-conf-file` *pathname*

  Specifies the path and filename of the 'login.config' file that tells mgetty which program to call for login. See the example login.config file to get some ideas what to do with it.

  The file name given will be ignored for security reasons if the file is not owned by 'root', or is readable or writeable by anybody else than 'root' (that is, it must be mode 0600 or 0200).

- `fax-id` [-I] *local fax number*

  This sets the fax station ID used in fax mode to identify your site to the caller (usually this is simply your fax phone number). Default is `FAX_STATION_ID`.

- `fax-server-file` [-S] *poll control file*

  Specifies the fax file(s) that is to be sent if someone else calls your modem in *fax polling mode*, that is, the caller *receives* a document.

  Normally, the file given is a text file, containing the list of G3 files to send to the calling machine, one file per line. Comment lines (starting with "#") are ignored. For backward compatibility, `mgetty` does check whether the named file is a G3 file itself, in which case this file is sent directly (but then, you can only send one page).

  Not all modems support fax poll *server* mode, I know that the ZyXEL and MultiTech do, and USR does not.

- `diskspace` [-k] *kbytes*

  This setting tells mgetty the minimum amount of disk space that has to be available in the fax spool directory for fax reception to be allowed. Default is 1 Mbyte.

- `notify` *mail address*

  This is the address that will get mails if a fax is received. Not fully tested.

- `fax-owner` *username/uid*
- `fax-group` *groupname/gid*
- `fax-mode` *perms*

  Similar to `port-owner/group/mode`, these settings specify the owner, group and file mode mgetty will use for incoming faxes. Defaults are taken from `FAX_IN_OWNER`, `FAX_IN_GROUP`, and `FAX_FILE_MODE`.

- `fax-spool-in` *dir1:dir2:dirn*

  Specifies a directory, or list of directories, where incoming faxes are saved. Multiple directories are tried in order until, the first one that has enough disk space and is writeable is used.

  The default setting is taken from `FAX_SPOOL_IN` in the Makefile usually `/var/spool/fax/incoming:/tmp` (/tmp is used as fallback).

- `debug` [-x] *debug level*

  This sets the amount of logging `mgetty` will do. A good value is *4*, more details are seen with `5`, and `9` is really noisy. Try it! The log data is written to the file specified by `LOG_PATH` in '`policy.h`', usually this is something like '`/var/log/mgetty.log.ttyxx`'.

- `gettydefs` *gd tag*

  If you use the gettdefs feature of `mgetty` – which is *not* recommended! – this specifies the gettydefs tag to use for the given line. See *man gettydefs*, *man mgettydefs*.

- `term` *terminal type*

  If you are on Linux or similar OSes that have `getty` set the `TERM=xxx` terminal type variable, and have no other method to set it (e.g. from '`/etc/profile`' or '`$HOME/.profile`'), `mgetty` can do it for you. Just specify '`term vt100`' or so. I don't think it's a good idea to specify the terminal type on a per line base (what if all your callers use different terminal types?), so the default is *unset*.

3 Fax Operations

Both mgetty and sendfax deal exclusively with raw "g3" fax files ("g3" stands for "group 3", which is the CCITT standard for encoding images for fax transmission). You will have to use external programs to create, view or print these.

There are two kinds of g3 files, the high resolution type with 204x196 dpi and the low ("normal") resolution ones with 204x98 dpi. If you mix up the two, the recipient will get a fax page that is either twice or half the proper length. You have been warned.

The width of a fax page is always 1728 pixels, the length is arbitrary (though there are some plain paper fax machines out there that limit the page length to A4 paper). A typical full page has a length around 2100 pixels in high-resolution mode.

3.1 Converting fax files

I recommend getting the '`pbmplus`' package written by Jeff Poskanzer, which includes lots of small programs to convert various bitmap formats into a portable intermediate format (`pbm`) that can easily be converted to fax format with the `pbm2g3` program. Further, it comes with lots of tools to scale, flip, and otherwise manipulate the `pbm` bitmaps. Be warned: it includes its own version of G3 conversion programs (`pbmtog3` and `g3topbm`), so be careful which one you use. The programs in the `mgetty` package (`pbm2g3` and `g32pbm`) behave slightly different (that is, they work!), and are significantly faster. Note that the `pbmplus` package does not include a graphical front end.

The '`pbmplus`' package can be found on most major FTP sites, e.g. on ftp.x.org in the '`/contrib`' directory. See Section 5.5 [ftp], page 60.

If you want to view the images using X11, you should get one of the many image viewers, like '`xview`', '`xloadimage`' or '`xv`'. See Section 3.6 [Fax-Tools], page 23. A simple, but very fast fax viewer can be found in '`mgetty/frontends/X11/`'.

Here are some examples for viewing fax files using `g32pbm`:

- You can print a fax on a PostScript printer (try '`lpr -Pps`' if you don't have '`lp`'):

```
cat $faxfile | g32pbm | pnmtops -noturn | lp -dest postscript
```

- or on an Epson-LQ, scaled for fine resolution (use -yscale 1.84 for normal resolution):

```
cat $file | g32pbm | pnmscale -xscale 1.76 -yscale 0.92 |\
    pgmtopbm | pbmtodot -360x180| lp -o epson -
```

- or you could view it using X11 via one of the following commands:

```
$ viewfax -v $file

$ cat $file | g32pbm >/tmp/fax.pbm ; xloadimage /tmp/fax.pbm

$ g32pbm $file | xv -
```

There are three easy ways to create g3 fax files, either use `pbm2g3` (included in this package. Do not use `pbmtog3` from the pbmplus toolkit. See Section 5.3.1 [pbmtog3], page 55.), use GhostScript (GNU Software, can be found on prep.ai.mit.edu) which contains a "digifax" driver that will produce the required format, or try Chris Lewis' '`hp2pbm`' package which will convert HP Laserjet print files into g3 fax files (`hp2hig3` program).

Once you have the right tools, there are lots of ways to create fax files for a wide variety of applications. Here are some examples:

- Ascii files can be converted using '`pbmtext | pbm2g3`' (use a *large* font, and don't convert texts longer than about 50 lines). Alternatively, you can convert ascii files to PostScript using '`psify`', the '`pslp.ps`' program from the ghostscript distribution, or similar tools, and feed their output into GhostScript.
- PostScript input can be converted by GhostScript (version 2.4 or higher), using the digifax ('`dfaxhigh`' and '`dfaxlow`') drivers. It will generate a plain g3 file with a 64 byte additional header, which will be skipped automatically. (You have to generate separate g3 files, one per page).

  A typical call to ghostscript would look like this:

  ```
  gs -sDEVICE=dfaxhigh -sOutputFile=/tmp/fax.g3.%d yourdocument.ps
  ```

  Do *not* use the "tiffg3" or similar drivers, they will create output files with headers that sendfax does not understand, thus causing the receiving fax machine to reject the data (it will assume that the transmitted headers are garbled data).

  If you use Ghostscript version 3.01 and up, you can use the '`faxg3`' driver as well, its output is identical to the output of the '`dfaxhigh`' driver except for the 64 byte header. Besides this, there should not be any difference.

  I have observed that with Ghostscript 5.01, the output of the '`faxg3`' driver is rendered better than that of the '`dfaxhigh`' driver. In addition, the former is compiled-in by default, while the latter is not. Thus, the default driver used by `faxspool` is now (starting with 1.1.7) the '`faxg3`' driver.
- Bitmaps can be converted using the '`pbmplus`' tools, but you'll have to do the proper scaling by hand. Use a pipeline with `pbm2g3` at the end.
- TeX dvi files can be converted to PostScript using '`dvialw`' or '`dvips`'. If you want to get the best possible output quality, you can use Metafont to create a set of 204x196 dpi fonts, which will look a lot better than scaled 300 dpi fonts (look into '`contrib/dvi-fax`' for instructions how to do this). You can use the `epsf` macros to include encapsulated PostScript files, e.g. a scanned signature.
- Another way for TeX file conversion is Ralf Schleicher's `faxdvi` package, found at '`ftp://ftp.leo.org/pub/comp/os/unix/networking/mgetty/faxdvi-1.1.tar.gz`'. Don't ask me about, ask him!
- HP-Laserjet files can be translated with Chris Lewis' '`hp2pbm`' package. It contains a program `hp2hig3` that will read HP-Laserjet '`PCL4`' files and produce G3 output.

  Warning: the G3 files that hp2hig3 emits lack the leading EOL code, thus causing `sendfax` to complain and possibly fail. As a quick fix, you can pipe those files through `g3cat`, it will fix the data.

A rather crude sample conversion program (`faxcvt`) is provided in the fax directory.

Better conversion, including guessing of the format of the input files, is done by the `faxspool` program, also provided in the fax directory. See Section 3.5 [Queuing], page 22.

3.2 Receiving faxes

If everything has been set up properly, faxes will be received automatically. Obviously, mgetty has to be listening to the proper modem line. Then, if a fax arrives, mgetty will store it in the directory FAX_SPOOL_IN (or the directory configured in 'mgetty.config', see Section 2.9 [runtime-mgetty], page 10) and send a mail to MAIL_TO (defined in 'policy.h').

The file name format is somewhat tricky (the reason behind it is that I have to create something unique without using mktemp(), because that would make it impossible to find out which pages belong to which fax). It is:

```
f<res><seq><line>[-<remote id>].<pagenr>
```

<res> is n or f, depending on the resolution of the fax. <seq> is a sequence number, 7 digits wide, somewhat related to the reception time. <line> are the last two letters of the tty device, e.g. 'S1'. If the sending side specified a fax station id, it comes next, after a leading dash (all blanks are replaced by dashes). Finally, after a dot, you'll see the page number.

If you want to process incoming faxes automatically, for example, print them, re-send them to another fax, send them by mail to you when you're on vacation, you could use what I call "notify programs":

If you define FAX_NOTIFY_PROGRAM in 'policy.h', mgetty will call this program (or shell script) when a fax has been completely received. It will be called with the following command line arguments:

```
FAX_NOTIFY_PROGRAM <hangup code> '<sender id>' <nr of pages> \
                   <file name page 1> <file name page 2> ...
```

<*hangup code*> is 0 if the receive was successful, non-zero otherwise. <*sender id*> is the fax identification string received from the other side. <*file name page (i)*> is the full path name for each received page.

A sample command line might look like this:

```
/usr/bin/new_fax 0 "+49 89 3243328" 1 /var/spool/fax/ff-01.a123
```

In addition, some environment variables are provide: CALLER_ID, CALLER_NAME, CALLED_ID (Caller ID and destination ISDN MSN, if available and supported by your modem), and DEVICE (the full name of the tty device, if you want to process faxes differently depending on the line they came in).

Such a "notify program" could print out the fax, convert it into a MIME metamail and send it away, display it in an X window (this a little bit tricky), or whatever. (A friend of mine uses it on his Linux box to call a program that will make the keyboard LEDs blink when a fax has arrived – as you can see, maximum flexibility is possible).

I provide a few examples (printing on HP laserjet, mailing as gzip'ed, uuencoded pbm file, ...) in 'samples/new_fax.*'

If you have the dialog shell tool, you can use the faxv program ("faxview") that I provide in 'frontends/dialog/faxv' to browse through all faxes in the incoming spool directory, view them, print them, rename and move them, ... 'faxv' is really more a sample program to show you how to do it, and you have to configure the X11 viewer and the printing program in the source code, but I use it for my faxes, and it works. Because of limitations on some operating systems, the list of faxes displayed is limited to the last 50 faxes received.

3.3 Basic `sendfax` usage

Sendfax is very primitive—no spool management, no format conversion, etc. It is designed to be able to send exactly one fax (consisting of multiple pages) to exactly one fax machine, but it is usable from within shell scripts. Its input are pages in "g3" format, either created with ghostscript or with pbm2g3.

It is called like this:

```
sendfax [-v] [-n] <phone-number> <fax page(s)>
```

e.g.

```
sendfax 0893243328 /tmp/to_gert_1.g3 /var/spool/fax/outgoing/picture.g3
```

It will then attempt to open the fax device and send all the pages to the remote fax machine, in the given order.

It will print little or nothing to stdout or stderr, except if it cannot find or open one of the named files, or if some transmission errors occur.

(There are a few stubs in the code to put headers on the pages at run-time, but since most class 2 faxmodems do not implement the command set properly, putting a header on a page does not work with them - because of that, I had to disable the corresponding code. `faxspool` works around this problem by using `g3cat` (see Section 3.6 [Fax-Tools], page 23) to "paste" a header line on top of each page)

If you specify '`-v`', sendfax will output more verbose progress messages.

If you specify '`-n`', it will try to send the fax data in normal resolution, default is fine resolution. (No conversion is done, so make sure that your input data is already in the proper format.)

Detailed reports can be found in the log file (usually '`/var/log/sendfax.log`') — but they may be of little more than technical interest since virtually all conversation with the fax modem is logged. (Nevertheless, if you send me any bug report, *please include all log files*)

Warning: Watch sendfax closely when sending the first few faxes. I had it abort a transfer a couple of times, not being able to recover and *not* hanging up the modem (the modem was completely locked up, with the phone line off-hook)! In my case, it was a problem of the modem that went away when I upgraded the firmware. Very old ZyXEL releases sometimes stopped the DTE and forgot to re-start it again.

The return codes of the sendfax program were chosen to make it easy for external programs (i.e. faxrunq) to decide whether to try again or not:

```
0: all pages transmitted successful
1: error on command line
2: cannot open fax device (typically happens if fax device is
   locked, but could also be a permission problem)
3: error initializing the modem
4: dial failed, "BUSY"
```

```
 5: dial failed, "NO DIALTONE"
10: dial failed, "ERROR" or "NO CARRIER"
11: waiting for XON failed (should never be seen)
12: transmitting page(s) failed (or polling failed)
15: some catastrophe hit (termination signal or blocked modem)
```

If the error code is less than 10, the attempt didn't cost anything, so an external program should try again to send the files. If it is 10 or higher, the failed attempt probably cost something, and the program should decide whether to try again (thus expending more money) or mail the operator and tell him something went wrong. My `faxrunq` program will suspend the job after five unsuccessful tries with error codes >= 10.

3.4 Fax polling using `sendfax`

Sendfax can also be used for fax polling (that is, you call someone and initiate a fax connection, and then *receive* a document. Don't confuse it with fax-on-demand, this is something different), the syntax is:

`sendfax -p <fax-number>`

or for sending a fax and then switch to polling:

`sendfax -p <fax-number> <send-documents>`

(in this case <*send-documents*> are sent, and then the documents from the other modem are polled, if there are any)

the received pages are written to the current directory, so make sure you have write access...

Warning: Fax polling does *not* work with ZyXEL Modems with ROM releases before 6.00 - with the 6.01 and 6.10 Eproms it works, but 6.11a is broken again. 6.12 and 6.13 work fine.

It definitely doesn't work with some Rockwell-based Faxmodems (Supra), since many versions of this chipset does not support polling. Some work, though, so you simply have to try it.

Could anybody try this with an Everex Faxmodem?

3.5 Automated fax queuing

For fax spooling and processing the queue, four additional utility programs are provided:

- `faxspool` will spool a number of files (accepting various formats) and put them into the fax spool directory. Its syntax is:

  `faxspool [options] phone-number input-file(s)`

  '`phone-number`' can be an alias, in this case the private and global alias files '`$HOME/.faxnrs`' and '`/etc/mgetty+sendfax/faxaliases`' will be searched for a line starting with this alias. The remainder of this line will then be used as fax number.

  `faxspool` interacts with the `file` program and '`/etc/magic`' to determine the file type of the input files. If your '`/etc/magic`' lacks entries for the various bitmap files, take a look at '`fax/etc-magic`', it contains the most important magic numbers.

`faxspool` reads the text to put on top of each fax page from a text file, '`faxheader`', located in '`/etc/mgetty+sendfax/`'. This file must contain pure ASCII text. Some tokens are replaced by the actual values: `@T@` becomes the destination phone number, `@P@` becomes the actual and `@M@` the total page number, and `@U@` is replaced by the user name. If your fax modem cuts of a few lines on top of each page, it may be a good idea to put a blank line before the header line itself, and it is also a good idea to indent the line about 5 spaces.

For the available options, please read the '`faxspool.1`' man page.

- `faxrunq` will read this directory and try to send all the faxes queued there (time scheduling is available, but primitive). If faxrunq succeeds, the fax is deleted and the sender is mailed. If it does not succeed after five tries ('`BUSY`' or a locked fax modem do *not* count for this) it will send a mail and not try any further to send this fax. (This should prevent your faxmodem from making you bancrupt. . .).

  `faxrunq` should be called at regular intervals from `cron` to process the queued jobs. It should only be executed by *root*, to make sure that all the files in the fax spool directory are read- and writeable.
- `faxq` will display all entries in the fax queue. If called with the '`-o`' parameter, it will also display completed, but not yet deleted jobs (JOB.done files). If called with the '`-v`' parameter, the output will be more verbose.

  Jobs that are just being sent are not shown (`faxq` doesn't see locked jobs)
- `faxrm` can be used to remove fax jobs from the queue. It is called with the job ids of the to-be-removed faxes as command line argument. The job ids are those that `faxq` returns.

These utilities do still have some rough edges, but I think they are fairly usable by now.

Earlier variants of those utilities had to be configured in the source code, from release 0.20 on, this is done by `make`. Please check the scripts nevertheless, whether all directories and files are set up properly.

3.6 Additional tools for working with g3 files

Some additional tools are provided for manipulating G3 fax files:

- `g3cat` (in the '`g3/`' directory) concatenates multiple G3 fax files. It accepts "raw" g3 files and digifax files and outputs a "raw" g3 file without headers. Its syntax is analogous to `cat`, except that you *have* to specify '`-`' to read from '`stdin`'.

  `g3cat` recognizes two flags: `-l`, to separate the fax files with a thin black line, and `-a` to byte-align the end-of-line codes in the file (Warning: some modems do not like this).
- To convert an incoming fax to Jef Poskanzers '`portable bitmap`' (`pbm`) format, you can use the `g32pbm` program that is also provided in '`g3/`'. It's advantages over Jef's `g3topbm` program are that it's approximately five times faster and takes only one eigth as much virtual memory (on machines with low virtual memory, it can thus be up to 100 times faster!). Its disadvantage is that it produces only "raw" (i.e., binary) '`pbm`' files; and it's not as stable when handling erroneous '`g3`' data (means that if one line is severly corrupted, it can happen that the next line will not be decoded properly. The rest of the file will then be OK again).

Syntax:

```
g32pbm [-r] [-s] [-l] [-d <dpi>] [g3-file]
```

If no g3 file is specified, standard input is used.

The `-s(tretch)` option causes `g32pbm` to double each row of the g3 file. You can use this to adjust the aspect ratio of a "normal" resolution fax file to match that of a "fine" resolution fax. Of course you can use other `pbmplus` tools for this instead, but `-s` is sure to be faster.

If you have a HP Laserjet printer, you can make `g32pbm` output HP LJ data with the `-l` switch. The desired resolution is chosen with the `-d 75|150|300` switch.

- For the other direction, I have provided a `pbm2g3` program that will convert Jef Poskanzer's Portable BitMap format into G3 fax data suitable for faxing with `sendfax`. The '`pbm`' files can be produced by various sources, e.g. drawing programs (`xpaint`), bitmap manipulation programs (`xv`), or simply be converted from other file formats by means of the '`pbmplus`' tools.

  As above, my variant is a lot faster than Jef's, and unlike his, it produces g3 data that adheres to the T.4 standard (if you don't fiddle too much with the options...), so you don't have to apply any patches to make it work together with `sendfax`.

  Syntax:

  ```
  pbm2g3 [-r] [-a] [-w page with] [-h blank lines] [pbm-file]
  ```

  if no pbm file is specified, standard input is used.

  For a detailed description, see the '`pbm2g3(1)`' man page.
- `viewfax`. To view G3 data files under X11, Frank D. Cringle has written a very nice and fast tool that will display the files on screen, turn them, zoom them in and out, . . .

  The source code can be found in the '`mgetty/frontends/X11`' directory. It's not (yet) built and installed by default, but it's straightforward.

  For a detailed description, see the '`README`' and '`viewfax.man`' files in that directory.
- To convert an incoming fax into X-Windows '`xwd`' bitmaps, you can use Chel van Gennip's `g3toxwd` program, found in the '`contrib/`' directory.
- If you want to print out faxes on a HP laserjet printer, you have two opions: you can use Chel's `g3tolj` program (also in the '`contrib/`' directory), or you use my `g32pbm` program with the `-l` option, which will make it output LJ data.

3.7 Using an external fax as a scanner

It is possible to tell mgetty to answer the phone even if it is not ringing (I call this "virtual rings"). Just send mgetty a signal `SIGUSR1`, this is usually done with '`kill -USR1 <mgetty-pid>`'. Mgetty will then pick up the phone and try to make a connection.

If you have a normal fax machine connected to the fax modem, it should be possible to have that fax machine dial any digit (to turn off the dial tone), and then have mgetty answer the phone to receive the "incoming" fax, thus using the fax machine as scanner without paying for a call. For a description of an sample setup (thanks, caz!), please see '`doc/scanner.txt`'.

Whether it works may depend on your phone company's setup, but it should.

If you have a modem that has a 'data/voice' button, it should also be possible to hit that button to make the modem pick up the phone. Mgetty will automatically notice that and handle it properly. (I've tested this only with ZyXELs - could somebody else please test this with other modem types and send me a logfile? Thanks)

3.8 Runtime configuration for sendfax: 'sendfax.config'

Basically, `sendfax` can work very well with the compiled-in defaults (from 'policy.h'), sometimes slightly modified by command line options.

If you use more than one fax modem, possibly of different type, this is very awkward, because the command line will get very long. For that reason, there is a configuration file which will control nearly every detail of sendfax' behaviour.

It is usually named '/etc/mgetty+sendfax/sendfax.config'.

The config file consists of multiple sections, separated from each other by the keyword 'port'. All configuration options given before the first 'port' statement specify global options, and everything between two 'port' statements applies only for the device with the given name ('port' takes an argument). Let me show you an example:

```
# global defaults:
# fax station id is always the same
fax-id ++49-89-1234
# always have the speaker on
modem-init ATM1L3

# port specific: for /dev/tty1a, switch the speaker off
port tty1a
modem-init ATM0L0

# port specific: for ttyS1, use another fax station id
port ttyS1
fax-id ++1234567
```

As you can see, empty lines and comment lines (starting with "#") are allowed.

Every line in the config file that is not a comment - or empty - starts with a keyword (listed in detail below), followed by a "data" field. In the example above, 'fax-id' is the keyword, '++49-89-1234' is the corresponding data, the fax ID for the sending machine. Most data fields are strings, but there are a few boolean and integer items.

The available keywords are (if it's possible to set this item from the command line, the flag is given in brackets).

– `fax-devices` [-l] *ttys*

  Sets the fax modems to use, e.g. 'tty1a:tty2a'. Default is `FAX_MODEM_TTYS` from 'policy.h'. The device names given here are used to look up the corresponding `port` section later. Watch out for upper-/lowercase.

  It is not very useful to specify this in the per-port section, so it is ignored if met there.

– `modem-init` [-m] *command*

  Specifies an '`AT...`' command that is to be sent to the modem right at the **beginning** of all modem talk (even before setting the modem into fax mode, so this could be an '`ATZ`' if you want to reset the modem).

– `modem-handshake` *command*

  Specifies an '`AT...`' command that will be sent to the modem at the **end** of the modem initialization, right before dialing. **Do not use ATZ or such here**, since resetting the modem will switch off fax mode. Default is `FAX_MODEM_HANDSHAKE`.

– `modem-type` [-C] *type*

  Force the modem command set to use. Default is '`auto`' (auto-detect, which may not work on very cheap modems), possible other values are '`cls2`', for "class 2 only" modems, and '`c2.0`' for "class 2.0" faxmodems. Default is `DEFAULT_MODEMTYPE`. See Section 2.9 [runtime-mgetty], page 10.

– `modem-quirks` *bitmask*

  Same as in `mgetty`, this can be used to adapt `sendfax` to some peculiarities in certain modems. See Section 2.9 [runtime-mgetty], page 10.

– `max-tries` *nnn*

  Specify the maximum number of tries per page if the receiving end reports reception errors. If *nnn* tries do not suffice to successfully transmit a page, `sendfax` will give up or simply go on, depending on the setting of `max-tries-continue` (see below). If this is set to '`0`', `sendfax` will ignore retransmission requests and simply go on. Default is `FAX_SEND_MAX_TRIES`.

– `max-tries-continue` *y|n*

  After the maximum number of tries for one page are reached, `sendfax` can report an error and abort (`max-tries-continue` *no*), or go on with the next page (*yes*).

  For "mission critical" faxes, I'd set it to *no*, but since the page quality is most often quite good even if reported as "page bad", the default is *yes*.

– `speed` *baudrate*

  Set the port speed to use for fax send operations. Usually, '`38400`' is a good choice, but a few dumb modems (for example, some based on rockwell chipsets) need '`19200`' or even '`9600`'. A few modems can go higher, but `sendfax` may not support it, and it may not always work. Default is `FAX_SEND_BAUD`.

– `switchbd` *baudrate*

  On some (very few) modems it is necessary to start with baudrate `"A"` and switch to baudrate `"B"` after sending the '`AT+FCLASS=2`' command. If you specify this, `sendfax` will switch from `speed` to `switchbd` right after setting the modem to class 2/2.0. Default is `FAX_SEND_SWITCHBD`, I'd recommend **not** using it unless you're sure that you need it.

– `open-delay` *msec*

  A few modems respond to raising the DTR line (when opening the device) with '`OK`'. This can confuse `sendfax`, because it will see this OK as response to the next command. In the log file, you can see if your modem exhibits this problem if the echo of each `AT` command can be seen when waiting for the response to the *next* `AT` command. With `open-delay` you can give a number of milliseconds to wait after the device is active until sendfax flushes all incoming responses and goes ahead.

The only modems that need this so far are an ELSA 33.6 and an ELSA MicroLink 56k, and delaying for about 1500 ms cured the problem.

– `ignore-carrier` *y*|*n*

  Some misbehaving modems lower the DCD (carrier detect) line briefly between sending multiple pages. Depending on the operating system used, this may cause all subsequent port accesses on this serial port to fail. You'll recognize it if you get a "weird-looking" sendfax log file that works fine up to the end of the first page, and then aborts with '`I/O error`' or so.

  In earlier versions, you could achieve this by setting `FAX_SEND_IGNORE_CARRIER` in '`policy.h`', but this has been removed. Use the config file instead.

  Modems where this is known to be necessary include all USR modems (Courier and Sportsters), some ZyXEL 1496 EG(+), and some GVC models.

  The default is `ignore-carrier yes`, as there are just too many users out there to read documentation, and it does less harm that way.

– `dial-prefix` *command*

  This is the command used for dialing out. Usually this will be something simple, as '`ATD`' or '`ATDP`', but if you have an unusual setup, it could also be something like '`ATX0DP0;X4DT`' (meaning: switch off dial-tone detection, pulse-dial '`0`', back to command mode, switch on dial-tone detection, and go ahead dialing with touch tones). The phone number will be sent right after the `dial-prefix`. Default is `FAX_DIAL_PREFIX`.

– `fax-id` [-I] *your fax phone*

  specifies the "fax station id" used to identify your fax machine to the receiving end. Usually this is the telephone number of your own fax machine, but if you want to (and if your modem support it), you can put up to 20 alphanumeric characters here, e.g. `+1-11-222 Fred`. Default is `FAX_STATION_ID`.

  Watch out: the `faxspool` program *only* uses the *global* definition from sendfax' config file, so the fax id it puts on the header might not be the same as the one transmitted by sendfax, if you use port specific settings for `fax-id` in '`sendfax.config`' (Paul Sands).

– `poll-dir` [-d] *full path*

  This is used to specify a directory where polled faxes (wheather maps and such) are to be saved into. Default is the current directory.

– `normal-res` *y*|*n* [-n]

  If set to '`yes`' or '`true`' (boolean), `sendfax` won't attempt to make a fax connection in "fine resolution" mode. Normally you won't need to use that option, since `faxrunq` will set the `-n` switch if needed. Default is '`no`'

– `fax-min-speed` *speed*

  Sets the lowest transmission speed that the modem will negotiate with the receiving modem. (Not implemented yet).

– `fax-max-speed` *speed* [-M]

  Sets the maximum transmission speed that the modem will negotiate with the fax receiver. Usually, you don't need this, as decent modems will pick the best speed automtically, but sometimes (e.g. for the USR Courier series) this doesn't always works, and transmission fails with +FHS:25. If you see that error, you might want to try `fax-max-speed 7200`.

– `verbose` *y*|*n* [-v]

  If set to '`yes`' or '`true`', `sendfax` will output progress reports on stdout, if set to '`no`', `sendfax` will only print error and warning messages. Default is '`no`'.

– `debug` [-x] *nn*

  controls the amount of information written into the fax log file (`FAX_LOG` in '`policy.h`'). '`0`' means "totally silent" (not even errors are written), '`9`' is really noisy. I usually use '`3`' or '`4`' in normal use, and '`6`' for debugging. Default is `LOG_LEVEL`.

– `page-header` [-h] *file*

  Yet unused (because of implementation shortcomings in all tested modems)

To show you how it will look like, I have included a sample '`sendfax.config`' file below. Three modem lines exist on my system, all modems have different initialization and flow control commands, and on one line I have to tell the modem to use pulse dial because the PBX is too old.

```
#
# sample configuration file for sendfax
#

# global settings
verbose y
# the modems are attached to the following ports:
fax-devices tty1a:tty2a:tty4c

# this is my fax number
fax-id +49-89-xxxxxx

# generic defaults
modem-init ATL3M1
dial-prefix ATD
debug 4

# port specific overrides

# Zoom 28K8
port tty1a
  modem-handshake AT&K4

# MultiTech
port tty2a
  dial-prefix ATDP
  modem-handshake
#                ^^^ this means "no extra command to set modem handshake"
  debug 9
```

```
# ZyXEL
port tty4c
  modem-init ATM1L2
  modem-handshake AT&H3
```

4 Voice Operations

This chapter explains how you can use mgetty to implement an answering machine if you have a modem that understands the voice command set (i.e. the ZyXEL).

The first version of these extentions was written by Klaus Weidner.

Since February 1995 these extentions are now developed and maintained by Marc Eberhard (Marc.Eberhard@Poseidon.ThPhy.Uni-Duesseldorf.DE). Since late 1998, Marc Schaefer (schaefer@alphanet.ch) took over vgetty development. Please see http://www-internal.alphanet.ch/~schaefer/vgetty.html for vgetty specific information, patches and releases, and please send email about the voice features to him, not to me.

I have removed the whole chapter from the documentation for now, as the voice stuff has changed too much, but the docs were not updated accordingly. Please see the programs and examples in the '`voice`' subdirectory for ideas how to get it going. Configuration is explained in '`voice.conf`'.

The '`voice`' subtree is **NOT** included in the official release 1.0, because of the lack of documentation, and because Marc thinks it's not stable enough yet. It *is* included in the 0.99 and 1.1 beta development trees, so if you want to play with it, get one of those version. **BUT** keep in mind what "beta" means: lacking documentation, problems, crashes, whatever.

5 Common problems and solutions (TROUBLESHOOTING)

This chapter tries to describe some known problems that can occur when installing or using this package. Don't worry, most of these have been solved.

5.1 Modems

This section describes problems that can occur when using various types of modems. Unfortunately, the class 2 fax implementations vary quite a bit between different manufacturers.

Many of the instructions for certain modem types below still refer to changing `#define`s in '`policy.h`' instead of listing the appropriate '`mgetty.config`'/'`sendfax.config`' lines, mainly because I haven't had the time to figure it out myself and didn't get enough feedback from config file users yet.

5.1.1 Problems common to many modem types

- Disable auto-answer in the modem (`ATS0=0`), because it will interfere with the manual answer done by `mgetty`. If you *have* to use auto-answer, set the `mgetty` ring-counter (`-n <i>`) high enough that `mgetty` will never try to answer the phone manually.
- Make sure that your modem is set to return *verbose result codes*, that is, set `ATV1`. Otherwise, the modem won't return '`CONNECT`' or '`RING`' but '`1`' or '`2`' as result code, and mgetty definitely doesn't understand that. By default, `mgetty` sets `ATV1` automatically, but if you change the init sequence, watch out for this.
- Make sure that your modem is set to the proper dialing type (`ATT&W` for touch-tone, `ATP&W` for pulse dialing) — `sendfax` uses `ATD...` to dial out, assuming that the modem knows which dialing method to use. Alternatively, set `FAX_DIAL_PREFIX` in '`policy.h`' or `dial-prefix` in '`sendfax.config`' accordingly.
- If the default modem initialization string in '`policy.h`', which is quite generic, doesn't work for your modem, you can either change it according to your needs, or store all settings in the non-volatile RAM (NVRAM) inside the modem and change the initialization strings to plain `ATZ` (modem reset).

  These settings ought to work with most modems:

  `ATS0=0`: do not auto-answer, `E1`: echo on, `Q0`: send modem responses, `&D3`: reset on DTR->low.

  The following, for example, are ZyXEL specific things:

  `&H3`: set handshake to RTS+CTS, `&N0`: "multi-auto" connect, accept all known protocols, `&K4`: enable v42bis/mnp5 compression.

  Naturally, you can use any init string you want (but the modem has to return OK) — check with your modem manual.
- On USR and some ELSA Modems, it has occasionally been reported that the initialization phase runs through fine, but incoming `RING` messages are not recognized, and mgetty complains about '`junk on the line`'. In all those cases, it were baud rate problems. The modem was initialized with an init-sequence of `"" ATZ OK` only. Problem

with this is, after ATZ, the modem changes to the DTE baud rate that the last AT&W command was sent with. If that baud rate differs from the `mgetty` port speed, a RING won't be detected. Fix is easy: send another `AT` command after the `ATZ`. For example, make the init sequence `"" ATZ OK AT OK`. The second `AT` command will set the modem back to the mgetty port speed.

- Some modems do not like the '`+FDCC=1,5,...`' command.

  In the latest version, mgetty and sendfax will automatically detect if the modem isn't able to do 14400 bps for faxing and will use 9600 bps — '`+FDCC=1,3,...`' — instead. So don't worry about the error message.

- Sometimes mgetty cannot initialize the modem, it times out waiting for the first '`OK`'. (USR Sportsters are known to hit this).

  I assume that the problem is the DTR-induced modem reset before sending the first '`+++ATH`' to the modem. I know that some modems need quite a lot time after a reset, so this should go away if you add more delays before sending the first string to the modem:

  In '`conf_mg.c`' (line 38 or so) change

  ```
  char *def_init_chat_seq[] =
            { "", "\\d\\d\\d+++\\d\\d\\d\r\\dATQ0V1H0", "OK",
  ```

  to

  ```
  char *def_init_chat_seq[] =
            { "", "\\d\\d\\d\\d\\d\\d+++\\d\\d\\d\r\\dATQ0V1H0", "OK",
                  ^^^^^^^^^ additional delays here
  ```

  Or, if you don't want to modify the code, just set the `init-chat` sequence in '`mgetty.config`' accordingly.

  Alternatively, you can just switch off the toggling of DTR (since the HUPCL flag is set, it should work as well) by setting `toggle-dtr no` in '`mgetty.config`'.

- Some operating systems send their commands too fast for some modems (Linux is known to do this). In that case it may be necessary to increase the delay times for `DO_CHAT_SEND_DELAY` and `FAX_COMMAND_DELAY` in '`policy.h`'.

- Some Modems toggle DCD between pages, possibly correlated to the presence of the "high speed" page transmission carrier. `sendfax` normally runs carrier-sensitive, and will then get a lot error messages in the `read` and `write` calls, logging them as `I/O Error` or something similar when sending the first or second page. To work around this, set `ignore-carrier true` in '`sendfax.config`'.

  If you run into this problem, `sendfax` will complain about '`Error +FHNG:-5 (Unexpected hangup / read() error / write() error)`' and abort.

- Some modems do not understand `ATH0`, responding with '`ERROR`' to that. Just modify the `init-chat` setting in '`mgetty.config`' to send `ATH` instead of `ATH0`. This will fix it.

- Faxing to some modems works very well, faxing to "normal" fax machines fails all the time.

  In most cases, this error is caused by badly created G3 files. If you use the old `pbm2g3` program, or an old copy of `hp2hig3`, it will cause bad G3 files.

Recently, a new problem appeared for postscript files created by certain programs that insist on setting their ideas of the page size (some versions of WinWord, FrameMaker, and dvipsk). If used together with ghostscript 3.*, the resulting G3 file's width isn't correct. For a fix, try creating the postscript file with a different program, use a newer version of ghostscript, or an older one (2.62 is fine). Quite often it helps to add `-sPAPERSIZE=a4` to Ghostscript's command line (in `faxspool`).

With a `mgetty` version later than June 1996, the `g3cat` program should take care of this problem, fixing the line width on-the-fly. It will print a warning, to tell you about the problem, though. So if you get this warning, faxing should work, but you'd better check all your programs anyway.

- Some other Modems cannot (or not properly) distinguish between calling faxes and calling modems.

  Some of them just refuse to answer both, only recognizing one and failing on the other (NO CARRIER) — in that case, there's nothing you can do to make it receive both (except making the manufacturer fix it). What you can do is to force it to receive always fax or always data (what is more important for you). Data-Only can be set by the `data-only` keyword in the configuration file, and Fax-Only with the `fax-only` keyword.

  In some cases, those problems may be caused by the modem forgetting the `AT+FAA=1` command if it receives any other command after it. To fix it, it may help to change the "answer chat command" in the config file to

  ```
  answer-chat "" AT+FAA=1;A CONNECT \c \r
  ```

  if this helps for your modem, please tell me so, and I'll include it into the documentation.

  Some other modems can distinguish the different call types *most of the time*, but some 2400 bps modems mysteriously fail. That may be caused by some – strange – modems sending a certain tone when *calling* the other side, and the receiving end mistaking that for the fax calling tone. Arne Marschall said about that: "...Or try calling your modem with your phone and whistle. If it says "+fcon" it is one one those which can't deal with modems using a calling tone" - try it.

  Another, quite simplistic approach of some modem manufacturers is that they distinguish Fax by waiting until the time specified in register `S7` (time to carrier) runs out, and then switching from data to fax carrier. That normally works quite well - if the other side is patient enough to wait that long . . . . For example, if the modem switches to fax after 60 seconds have passed, and the caller has a timeout of 50 seconds, it will definitely fail. If in doubt, try setting `ATS7=30` (but only if all else fails - and don't ask *me* why it doesn't work...)

- Except if you feel sure that your modem needs less, do not try to send or receive faxes with a port speed of less than 19200 bps. Since fax transmission is synchronous (no start and stop bit!), you need at least 12000 bps on the Computer-Modem line to transmit 9600 bps on the line.

- For some modems, incoming faxes are detected properly, but mgetty times out about two minutes later in `fax_wait_for(OK)`. Most likely, the `switchbd` ('`mgetty.config`') speed is set wrongly. A number of cheap modems (e.g. Rockwell-Chip based ones) step to 19200 bps upon fax reception, and setting this option to '`19200`' will tell mgetty about it. Better modems have no need for that, so if you have defined it and your modem does *not* change bit rates, it won't work either.

If you have this problem, your log file for an incoming fax call will look like this:

```
09/21 09:55:48 yS1  waiting for ``RING''
09/21 09:55:48 yS1  got: [0d][0a]RING ** found **
09/21 09:55:48 yS1  send: ATA[0d]
09/21 09:55:48 yS1  waiting for ``CONNECT''
09/21 09:55:48 yS1  got: [0d][0a]ATA[0d][0d][0a]FAX
09/21 09:55:50 yS1  found action string: ``FAX''
09/21 09:55:50 yS1  start fax receiver...
09/21 09:55:50 yS1  fax receiver: entry
09/21 09:55:50 yS1  fax_wait_for(OK)
09/21 09:55:56 yS1  fax_wait_for: string '+FCON'
09/21 09:57:50 yS1  Warning: got alarm signal!
09/21 09:57:50 yS1  fax_read_byte: read returned -1: Interrupted system call
09/21 09:57:50 yS1  fax_get_line: cannot read byte, return: Interrupted system call
```

- For a couple of modems, you can find initialization and setup hints in the file '`doc/modems.db`'. Just browse through it, maybe you'll find something helpful. (You're welcome to provide entries for modems not yet in the list).
- Sometimes, very weird things happen upon login (that is, long after `mgetty` has done its work), for example, /bin/login blocks infinitely, the caller is thrown out immediately again, or the shell won't notice a logout, etc.

  Quite often, this is caused by a modem not setting the data carrier detect (DCD) line properly (properly meaning here "reflecting the actual line conditions"). On most modems this is done with the `AT&C1` command.

  Another possibility are too long or otherwise broken modem cables that corrupt the signals, and introduce noise on the modem control lines. Flat cables are famous for this, use round, shielded modem cables, no longer than necessary.
- `mgetty` will complain loudly if you switch off your modem.

  Well, what should I say, this is a feature :-) – I think that mgetty should detect and complain if a modem is not working, and if you switch it off, it is most definitely not working.

  For most modems, there is an easy way out. `mgetty` can query the modem's DSR line at startup, and wait until that line comes active. For that to work, you need to set DSR to "always on" with `AT&S0&W` (entered from a terminal program), and then switch on `need-dsr yes` in '`mgetty.config`'. Check the log file (with `debug 6`) to see whether it's working. It won't work for old unix systems where you can't query the state of the serial port control lines (DSR, DCD, etc.).

  For modems or unix systems that have problems with DSR, there's another approach: You just create a file, '`/etc/nomodem.ttyS1`', or however you want to name it, when you intend to switch off your modem. Instead of running `mgetty` from '`/etc/inittab`', you run a "wrapper" script that will wait until the nomodem file disappears, and then starts `mgetty`. That way, you can just prevent `mgetty` from starting when you don't want it to.

  The wrapper is easy:

```
#!/bin/sh
#
while test -f /etc/nologin.ttyS1
do
   sleep 60
   done
exec /usr/local/sbin/mgetty ttyS1
```

- I have been asked about "superfine" fax mode recently (using 392/408 dpi instead the usual 198/204 dpi). It is not yet implemented. It would not be not overly hard, but you need a modem that supports it - no class 2 modems that I know of support it, and only very few 2.0/2.1 modems (MultiTech). So it hasn't been done yet.
- Similar to the last bullet, V34 fax (using transfer speeds up to 28.800 bps) isn't fully implemented yet. My modems do not support it, and nobody has asked for it yet. It should be really easy, though.

5.1.2 ZyXEL

First of all, yes, ZyXELs tend to somewhat non-deterministic behaviour - most modems work perfectly with `mgetty+sendfax`, a few don't, and I do not always find out what's wrong.

If it works first, and then suddenly stops working, it does quite often help do to a **full** modem reset, that is, switch it off, press the DATA/VOICE key, switch it on again and hold the key for about 20 Seconds. (That's for the 1496E and E+, for other models, it works differently, check you manual). That cured the problems in many cases.

The same holds true after a firmware change: always do a full modem reset in case you change the ROM version!

Do not use non-plus ZyXELs (the "slower" ones) for faxing with DTE speeds above 38400 bps, I got a report that it won't work.

Do not use the `S18` register, set it to 0. Its purpose is for dumb `getty` programs where the modem has to change its baud rate back to a certain value if some other program has modified it. Since `mgetty` will reinitialize the modem anyway if another program has dialed out, the `S18` register cannot improve anything, it can only harm (e.g., if it is set to `3`, the modem will send any `RING` with 19200 bps — imagine what happens if `mgetty` sets the port to 38400 . . .)

If you want to use Caller ID (and have subscribed to it), add S40.2=1 to the modem initialization string [For the ZyXEL 2864, this might be S84.4=0, but check with your modem manual!].

Warning: If you use a very old ZyXEL and try to send some complex graphics (the "tiger.ps" example file in the GhostScript package is known to produce this), it may lock up. This is because the old ZyXEL firmware had *lots* of bugs concerning hardware flow control—sometimes the ZyXEL just forgot to raise CTS when it can accept data again. The symptoms are that the transmission hangs, no modem LEDs are flashing any more, the logfile won't grow any more and the receiving machines hangs up after printing a partial page.

This bug has been fixed in ROM release 6.01, so you should consider upgrading your eproms if you run into this. With ROM release 6.01 and 6.11a, it works correctly (a hundred-odd faxes sent without problems).

Rom releases I've never been able to make `sendfax` work reliably are 6.00 and 6.10 (6.10a works), and various 5.0x releases.

Fax polling (both server and client side) is broken in ROM 6.11a.

The very latest ROM releases, 6.12, 6.13 and 6.15, work perfectly in every possible aspect. I tried fax sending, receiving, polling, data calls, . . . - and everything simply *works*!

The latest ZyXEL roms can normally be found on ftp.zyxel.com, in '`/pub/other/zyxel/r*`'.

Some models, seemingly only the '`EG+`' (german telekom approved), toggle the DCD line during the pre-page handshake, thus causing `sendfax` to abort (with a "I/O error" during read/write operations). You can work around this by defining `ignore-carrier true` in '`sendfax.config`'.

ZyXEL modems can do security callback (`AT*Gi` with `i >` 0). With ROM releases up to 6.11a, this doesn't interact very well with mgetty. If you want to use it anyway, send me a note and I'll describe what to change (but be warned, mgetty operation will become somewhat unreliable by those changes!). With 6.12, it works very good.

Oh, one additional note: ZyXELs can do voice recording and playback. Klaus Weidner and Marc Eberhard have written voice handling tools, to turn your modem into a sophisticated answering machine. Yo can finde this tools in the '`voice`' subdirectory. (NOTE: In version 1.0, I haven't included the stuff, because the docs are heavily lacking and Marc thinks it's not ready yet. Get 0.99 or 1.1-betas to play with voice).

Josef Wolf (jw@raven.inka.de) has added some comments about the ZyXEL Eproms (I don't know whether it's true or not, ask him or support@zyxel.com):

Most EPROMs require /PGM and Vpp to be Vcc while in ROM-Mode. On ZyXELs these two pins are removed in the socket so the pins are _floating_. This treatment is **out of spec** for *any* EPROM. The TI-Types originally used by ZyXEL seem to tolarate this treatment. But most other EPROMS won't work properly. There are two solutions to this problem:

1.) use the types which zyxel used. (could be troublesome to find some)

2.) take two sockets, solder jumpers between pins 1, 31 and 32 (these are /PGM, Vpp and Vcc) and place them into the original sockets before inserting the EPROMs.

NEWS: ZyXEL has a new modem line out in the market, the *ZyXEL COMET 336* modem. This is a fairly standard Rockwell-based junk modem. It can't do class 2 or class 2.0 fax. Don't buy it.

5.1.3 Telelink IMS 08 Faxline+ Modems

Thanks to friendly support by MVS, Germany, I got a Telelink IMS 08 modem for testing and was able to adapt `mgetty` accordingly.

The modems factory defaults are very good, so it's sufficient for `mgetty` operations to set `init-chat` (in '`mgetty.config`') to *ATQ0V1E0 OK AT&F2S0=0&D3X6&K3 OK*, and `switchbd` to *19200* (yep, it switches baud rate. Stupid, but true). After that, receiving fax + data calls works fine.

Fax sending is not that trivial. Basically, it works after setting `modem-handshake` to `AT&K3`. `FAX_SEND_FLOW` (in '`policy.h`') can be anything that the host machine supports (because the modem does both Xon/Xoff and RTS/CTS handshake simultaneously).

Unfortunately, a few problems remain:

– faxing multi-page faxes to ZyXELs with ROM release 6.11a doesn't work (fails on the receiving end with +FHNG:100 and on the sending side with +FHNG:50).

– faxing to some paper fax machines fails completely, for others, "only" complex graphics (like the Ghostscript-"tiger.ps") fail. This one can be partially cured by adding lots of padding zeroes into the fax data (`g3cat -p 24`) - but that's unelegant, not complying to class 2 specs, and not supported (besides, it still doesn't work all the time). Maybe later versions of `sendfax` will do the padding automatically.

I recommend against using this modem with sendfax. In addition to the technical problems, their customer support (at least, that of the german distributor MVS) is basically non-existant. (I wrote them four times describing my problems with the modem, and never got an answer).

5.1.4 Rockwell-based modems, e.g. Supra

As far as I know, sending or receiving are no problem (although you have to use 19200 baud when in class 2 faxmode - set `speed` to `19200` in '`sendfax.config`', and set `switchbd` to `19200` in '`mgetty.config`'. Remember to change the modem initialization strings to the proper values for your modem, that is, *ATQ0V1E0 OK AT&K3S0=0 OK*.

Especially for the '`SupraFax`' modem, I've been told that you have to set `modem-handshake` to `AT&K3` and initialize the modem with `AT&S0&D2&K3`. Since the modem does not like being reset with DTR->low (do **not** use `&D3`!), an `ATZ` in the first initizalization string in '`mgetty.c`', to reset the modem into a known state, is a good idea, too.

(Thanks to Christof Junge, chris@cj.in-berlin.de, for trying out several weeks until everything worked!)

Some other SupraFAX Rom releases seem to forget that they are set to use RTS/CTS handshake if a `+FCLASS=0` is sent to them. I think it should help to store the `AT&K3` into NVRAM, but maybe you have to patch '`mgetty.c`'. See '`contrib/readme.supra`' for details.

Fax polling does not work because the Rockwell chipset does not support it.

NEWS: Most recent Rockwell 33.600 modems do not support any decent fax operation anymore – they added "simultaneous Voice over Data" (SVD) to their modem firmware, and because the Eproms are not large enough, they threw out their class 2 firmware. Don't buy such a modem, it won't work properly with `mgetty+sendfax`.

If you buy a Rockwell-based modem (they are usually quite cheap), make sure that you get one that can do class 2 or (better) class 2.0. Even if it's written on the box, some recent models just can't do it!

Together with `vgetty`, many Rockwell modems can distinguish between different types of incoming "RING" tones (usually called "distinguished RING" by the local Telco). Use the command `AT#SDR=n` (n between 0 and 7) to enable this feature. If in doubt, check with your modem manual whether your modem can do this at all.

5.1.5 Zoom VFP/VFX 24K FaxModem (V.FAST modem, 24,000 bps)

For the Zoom V.FAST 24,000 modem, you should change `init-chat` to `ATE1Q0V1 OK AT&FS0=0&C1&D2W1 OK` (see the manual for the meaning of the commands). After that, everything should work. (I got very euphoric reports).

My own experience with my Zoom VFX 28.800 is also quite good, but it doesn't seem to like automatic fax/data distinction too much, that is, some data calls are "recognized" as fax and fail miserably. Dunno how to fix it, I run my Zoom as data-only.

Fax polling doesn't work.

5.1.6 Best 14496 EC fax modem

Works quite well. Use `FLOW_HARD` for all flow control settings (in '`policy.h`', use 19200 bps for sending and receiving. Set `modem-init` to `ATE1Q0V1 OK ATS7=60&D3\\Q3%C1\\N7 OK` ('`mgetty.config`'), and `modem-handshake` to `AT\\Q3&S0` ('`sendfax.config`').

After that it should work. Kudos to Sami Koskinen (tossu@krk.fi).

5.1.7 GVC FM-144Vbis+/1 (Rockwell-based)

Basically, the modem is similar to the SupraFax modem.

Change `speed` in '`mgetty.config`' and '`sendfax.config`' to `19200` and set `modem-handshake` to *AT&K4*.

Further, if your model toggles DCD betwen fax pages (`sendfax` fails mysteriously between pages), `ignore-carrier true` in '`sendfax.config`'.

After that, it should work.

Note: If your modem doesn't properly distinguish incoming fax from data calls (i.e., a 2400 bps caller is repeatedly "identified" as fax caller), upgrade your firmware. I've been told (thanks to John Watson, watson%carssdf@rutgers.edu) that a new firmware release, **v1.69** exists that will fix those problems.

5.1.8 CREATIX Modem (Rockwell-Based)

For the new CREATIX modem, use the following settings in '`mgetty.config`' (thanks to Jens Hektor, jens@physiology.rwth-aachen.de):

```
speed 38400
init-chat "" ATE1Q0V1 OK ATS0=0&D3&K3
```

and `modem-handshake` *AT&K3* in '`sendfax.config`'.

The modem has a voice mode, too, and it should work with `vgetty` by now. As the docs for vgetty are out-of-date, I don't really know how to get it to work, though.

5.1.9 German Telekom approved GVC modems

(GM-144VBIS+ RC9696/14 (?))

This modem does *not(!)* use Xon/Xoff flow control. Further, the default modem profile sets '`&S1`', which makes the modem disable DSR all the time. On systems using the FAS serial driver, this will *disable CTS flow control*!

So, #define `FAX_MODEM_HANDSHAKE "AT\\Q3&S0"` in '`policy.h`', do not define `FLOW_SOFT` for flow control, and fax sending should work. (It does for me!) Changing `FAX_SEND_BAUD` to `B19200` is not necessary, it works with `B38400`.

Fax receiving... I did not fully test it. It's somewhat difficult since that modem insists on using auto-answer, but it should be possible to let it auto-answer if you set mgetty's ring counter high enough. Or, you can trick the modem, by changing mgetty's answer command (`ATA`) into `ATS0=1` – upon the next RING, the modem will "auto-answer".

5.1.10 Dallas Fax 14.4

Change `FAX_SEND_BAUD` and `DEFAULT_PORTSPEED` to `B19200`, change all occurances of '`AT&H3`' to '`AT&K5`', remove '`AT&N0`' and '`&K4`' in '`policy.h`'.

5.1.11 Everex

All I programmed is strictly to everex specs, thus, it should work. Most likely, some fiddling with the initialization strings is necessary. (If you have an Everex modem, please report to me what you did change).

5.1.12 Exar 9624 fax modem

This modem needs two stop bits (when sending), so you have to add `CSTOPB` to `tio.c_cflag = ...` in '`tio.c`' / `tio_mode_sane()`.

Also, use `#define FAX_SEND_BAUD B19200` and, for fax reception, `#define DEFAULT_PORTSPEED B19200`.

Further, the modem is a little bit timing critical - please insert `delay(500)` calls at the end of `fax_open_device()` ('`sendfax.c`', line 122) and before sending the page punctuation (`fax_command("+FET=...")` in '`sendfax.c`', `main()`, around line 540).

Also, for at least one Exar 9624 (built into a Compaq notebook), it's been necessary to add two delays (`\\d\\d`) before the `AT+FCLASS=0` initialization string in '`mgetty.c`'.

5.1.13 Tornado / Lightspeed modems

Here is a suggested setting for the default profile for these modems. See Section 5.1.1 [Common], page 31.

For Lightspeed store profile:

```
at&f
at&c1
at&d3
ats0=0
at%c2
atw1
at&w
```

and for tornado store profile:

```
at&f
at&d3
ats0=0
at&w
```

Then just initialize the modem with `init-chat` set to `"" ATZ OK`.

5.1.14 Zoltrix Platinum Series 14.4

This modem is also Rockwell-based, so don't expect anything unusual... - it works quite well, both fax sending and fax/data answering. You should use the following settings in '`policy.h`' (suggested by las@io.org, Laszlo Herczeg)

```
#define DATA_FLOW FLOW_HARD
#define FAXREC_FLOW FLOW_SOFT
#define FAXSEND_FLOW FLOW_SOFT
#define FAX_SEND_BAUD B38400
#define FAX_MODEM_HANDSHAKE "AT&K3"
#define MODEM_INIT_STRING "ATS0=0V1Q0&D3&K3%C1W1"
```

In some circumstances, sending of **large** graphics files (the well-known '`tiger.ps`') may fail.

5.1.15 MultiTech modems (MT1432BG and MT2834BG)

First of all, I want to thank MultiTech Munich for their support (borrowing me two test modems w/o charge).

Second, I can only strongly recommend these modems, they are **great** (but expensive). Got it, unpacked it, switched it on, set mgetty's init string (`MODEM_INIT`) to `ATS0=0&S1` – and everything simply worked. Flawlessly. (Warning: usually I recommend `AT&S0` to force the DSR line high – needed, because otherwise some O/Ses won't do hardware flow control – but that doesn't seem to work on my model. AT&S1 means that H/W flow control only works when a carrier is present, but then, who needs flow control in command mode?)

The modems do very reliably distinguish incoming fax and data calls, and outgoing fax works also very good (unfortunately, it's limited to 9600 bps fax rate, but that's no big problem).

The only problem I've seen is that those modem do the fax bit order on reception **right** (everybody else does it wrong, to be compatible with Rockwell, who botched it in the first place). Thus, `g32pbm` won't be able to decode the data, unless you call it as `g32pbm -r`.

(You can see if the bit order is wrong if g32pbm complains in every single line). I'll work something out for one of the next releases to work around this (*modem-quirks 02*).

BUT: There seems to be a way to tell the modem to behave like a Rockwell one and use the "wrong" byte order. Carlos Fonseca found the following text on ftp.multitech.com:

```
Function          Command       Description

PROCESS           +FRBOR        Maintaining compatibility with
DATA IN                         Rockwell Class 2 chip set for fax data
DIRECT OR                       reception .
REVERSE ORDER                   FRBOR=0 - Process received fax data in
                                direct order.
                                FRBOR=1 - Process received fax data in
                                reverse order.
```

so, with AT+FRBOR=1 added to the modem initialization string, it should be possible to get fax reception on the MultiTechs going without tricks.

Glenn Thobe suggested the following definitions for '`policy.h`' (which mostly are factory defaults anyway, but it wouldn't harm to set them)

```
#define MODEM_INIT_STRING "ATS0=0Q0&D3&E1&E4&E13&E15"
#define FAX_MODEM_HANDSHAKE "AT&E4"
```

My '`mgetty.config`' for those modems looks like this (everything not mentioned is set to the defaults I ship in '`policy.h`').

```
# MultiTech MT1432BG
port <whatever1>
  init-chat "" \d\d\d+++\d\d\dATE0E1V1H0 OK ATL0M0S0=0 OK
  modem-type cls2

# MultiTech MT2834BG
port <whatever2>
  init-chat "" \d\d\d+++\d\d\dATH0&F OK ATX4&D3 OK ATQ0L0M0S0=0 OK
  modem-type cls2
```

Some of the newer 56k-capable MultiTech modems have voice functionality, but some firmware versions are very much broken. Others seem to work more or less. So make sure you can return your modem to the dealer if it doesn't work - and if the dealer isn't willing to do this, get a different modem. Russell King told me that the firmware version 0316D should be working ok, but I got some negative reports as well. Hmmm.

News The most recent series of MultiTech modems (MT5634ZBA-V92 external, or MT5634ZPX-PCI-V92 / ISI5634PCI internal), based on the Lucent/Agere Venus chipset, seem to be really good for fax, data and voice support. Older firmware releases had bugs in the voice area, but if you use the most recent firmware version, they should be fine. For vgetty, you might need to set "forceV253 true" and "rec_compression 6" in '`voice.conf`' (Contributed by Lee Howard, <faxguy@howardsilvan.com> – thanks).

5.1.16 ELSA voice/fax modems

ELSA makes a nice series of Data/Fax/Voice products, both for POTS lines and for ISDN. In general, all of them are supported fairly well by `mgetty+sendfax` (fax works, fax polling works(!), voice works), but here are a couple of notes for different products:

- MicroLink 56k/56k pro – standard analogue voice/fax/data modem. Pretty good for about everything, but make sure you use the latest firmware version (1.58), as earlier versions have bugs switching between voice and fax mode.
- MicroLink Office – in addition to being a nice modem, this thing can do fax reception and answering machine functionality in "standalone mode", without a computer. Received voice calls and faxes can be downloaded via X-Modem to the computer. I'm working on a nice GUI application to ease using the standalone mode. The ML Office seems to be the only non-ISDN ELSA modem that can do Caller ID (CLIP).
- MicroLink TL/V.34 – this is an ISDN terminal adaptor that can also do V.34 modem calls and fax. Works nicely.
- TanGo – this an ISDN only terminal adaptor. Works, but it's pretty boring stuff.
- MicroLink 28.8/33.6 TQV – this is the old voice/fax/modem series. Works, but uses a different command to got to hardware flow control mode (`AT\Q3` instead of `AT+IFC=2,2`) so you'll see some errors in the voice log files. Don't worry about that.

If you have any problems with ELSA modems and mgetty concerning fax/voice support, report them to me first, and let me report them to ELSA - I'm actively working together with them to iron out bugs, and it's easier if all the reports come from only one contact person (me).

5.1.17 US Robotics (now 3com) Courier/Sportster Fax/Data modems

There are a number of major lines of 3com/US Robotics modems:

- USR Courier V.34/V.everything – those are quite expensive, but really, really good modems. I think those are the best V.34/X.2/V.90 modems around, and they are also very good for class 2.0 faxing. But make sure that you use a recent firmware, as older versions had very bad (and stooopid) bugs in the fax department. You can find out the firmware version with the `ATI7` modem command. The '`Supervisor rev`' version listed should be at least '`6.5.3`', better '`7.1.8`' (this is the '`X.2 upgrade`').

  Fax polling still doesn't work, and if you send faxes to a modem that can only accept fax with 7200 bps, it will fail with `+FHS:25` unless you specify `sendfax -M 7200`.

  The Courier series does not have voice features.
- USR I-Modem – this is an ISDN Terminal Adaptor with built in V.34/fax mode ("ISDN Modem"). Nice product. Works. The same stuff said for the Courier V.34/V.everything is true for the I-Modem.
- older USR Courier modems – older series without flash ROM, and possibly not upgradable to V.34/X.2 at all, might give problems. Fax was added fairly late, and the first fax-capable releases are very buggy. If you have such a modem, and `ATI7` does not list

‘V34’ in the ‘Options’ line, you might try the FAX_USRobotics option in ‘policy.h’. Maybe it helps. But don’t expect too much.

- USR Sportster series – the “Sportster” is actually a range of many different low-end modems. Most of them can do voice stuff, some of them have flash ROM to be upgraded to newer firmware. For data, this modem is quite good. For fax, it might work, and it might not – I have received a very high number of “it doesn’t work at all” reports, and also a number of “it works very well for me” reports. If you plan to buy one, make sure you can return it if it doesn’t work.
- Internal PC card – I haven’t been able to find out yet whether that is a “Sportster” or a “Courier”, but it seems to be a Sportster, with all its negative habits.

Some older USR firmware versions had severe bugs when doing RTS/CTS (that is, hardware) flow control. Occasionally, a byte got lost, and sometimes this confuses the modem enough to abort transmission with the error message

```
Transmission error: +FHNG:44 (Unrecognized Transparent data command)
```

Sam Leffler recommends using Xon/Xoff flow control for both fax sending and fax receiving (`#define FAX_SEND_FLOW FLOW_SOFT` and `#define FAX_REC_FLOW FLOW_SOFT` in ‘policy.h’).

Some day in the future, I’ll make those "quirks" configurable from the config file, but until then, you’ll have to live with recompiling. Sorry. (Upgrade your firmware!).

Fax polling with the USRs is *not* working, even though the modem claims so. It will work half-way through, but fail miserably later.

When sending faxes with an USR faxmodem, set `ignore-carrier yes` in ‘sendfax.config’. Otherwise it will fail after the first page with a read error and error code -5. (But that is default anyway, if you use mgetty 1.1.16 and up).

For some fax receivers, a problem remains: the USR modems do not want to negotiate 7200 bps transmission speed. If the receiving modem won’t accept 9600 bps, transmission will fail after sending three DCS frames, with the error code +FHS:25. In that case, try setting `fax-max-speed 7200` in ‘sendfax.config’.

The USR modems support Caller ID delivery (both the I-Modem series and the analog ones). Switch it on with `AT#CID=1` and mgetty should automatically be able to recognize the incoming caller ID. If not, show me a detailed log file...

Firmware upgrades and documentation are available on `ftp.usr.de` in the directory ‘/pub/USRobotics/modem/...’.

5.1.18 Elink ISDN Terminal Adaptors 293, 310, 393 with X.75 and V.110

The TA’s are connected to national German ISDN (1TR6) or Euro-ISDN. The host side is a standard serial port with an AT-command set, letting them look like a conventional modem. Therefore they are often (wrongly) called ’ISDN modems’.

It is strongly recommended to feed them with 115k2 bps, else only V.110 (38k4 bps) is available. Configuration may differ slightly, depending on which of those Elinks is used, wether it is connected to Euro-ISDN or 1TR6 and last but not least they are still under developement, so you’re on your own with that.

ISDN generally supports two nice features: first it is now possible to check callers number, which may be used to identify callers, second ist the charge service, where it is possible to request the amount of charge units for the call. For mgetty the second one is only from minor interest, but the first one is. The opposite phone number will be shown with the command `AT\O`. If a call comes in, `AT\OA` will answer the call, display caller's id in a 'digit only' (e.g. '`04419808550`') form and then print out the `CONNECT`-String.

To enable mgetty to utilize this, the "get caller ID" sequence must be set up accordingly and the '`CND`' feature must be enabled:

Use `get-cnd-chat "" AT\\O OK` in '`mgetty.config`', and make sure that `#define CNDFILE "../dialin.config"` at the end of '`policy.h`' is enabled.

If you only want to grab the phone number and not check it against '`dialin.config`', you can try playing with something like `answer-chat "" AT\\OA CONNECT \c \r` in '`mgetty.config`'.

Having a glance at the output of `AT I4`-output of the Elink, it looks as if it is able to support Fax-Service too, but there is no hint in the manual. So mgetty will likely put itself into data-only mode. (I got some information from the Elink people that some of the Elinks have a data/fax analog modem built in, which should work nicely with the fax part of `mgetty`, but I didn't try yet.)

(Thanks to Ulrich Tieman, lord@olis.north.de, for this section. Don't ask me, ask him if you use an ELINK)

5.1.19 ISDN 4 Linux support

`mgetty` and `sendfax` work with isdn4linux (i4l), but there are a number if things to point out.

For data calls via ISDN V.110 or X.75 async modem emulation, you can put mgetty on a '`/dev/ttyI*`' device and mgetty will handle so-called "asynchronous ISDN" connections fine. To make sure that mgetty will pick up the phone, set your local ISDN number with `AT&E<number>` from `init-chat` (otherwise isdn4linux will just ignore incoming calls).

As far as I can see, i4l will always signal the caller ID for incoming calls as "CALLER NUMBER: ..." to mgetty. Mgetty can handle this just fine.

If you want to see the destination number for incoming calls, add `ATS23.2=1` to the `init-chat`. This will make i4l signal the dialed MSN number as `CALLED NUMBER: 12345`, and mgetty will then pass it on as $CALLED_ID to fax scripts and so on. This is implemented since Linux 2.4.10, or in earlier Versions with Jan Oberlaender's patch (http://www.stud.uni-karlsruhe.de/~uslk/i4l-callednumber.tar.gz).

Alternatively, you can set `ATS23.0=1`. This will make i4l signal the dialed MSN number as `RING/12345`. Unfortunately, it will be recognized by mgetty as caller ID, not destination MSN, so you might need to do some fiddling with 'ring.c' to get this fixed.

For the full list of available AT commands, consult the i4l docs (for example in '`/usr/src/linux/Documentation/isdn/README`'.

For ISDN PPP calls with synchronous PPP (which is the "normal" way to do PPP over ISDN), you do **not** need `mgetty`. PPPoISDN calls are handled by `ipppd`, which is part of isdn4linux. Mgetty can only handle PPP-over-X.75 or PPP-over-V110, which is something that is only used for WAP connections or with old terminal adaptors.

Fax support with isdn4linux is a tricky issue. For most ISDN cards, this will **NOT WORK**. It only works for cards that have a fax chip (or separate CPU) on board, and can be accessed with the fax class 2/2.0 command set. It will work fine with the Diehl Server BRI ISDN card (active), and that's the only card I know that will work as of June 2001. Accessing ISDN fax via CAPI is NOT supported.

There's a small catch: modem autodetection will make mgetty think that your ttyI device doesn't support fax (because most cards can't do it). So you need to set `modem-type c2.0` in '`mgetty.config`' and/or '`sendfax.config`' to force class 2.0 mode.

Voice support with `vgetty` should work (use your ttyI device as an answering machine, etc.). I have not tried it yet - talk to the mgetty or i4l mailing lists if it doesn't work.

5.1.20 Class 1 Faxmodems

These do not work. They are not going to be supported (class 1 faxing is a mess, and the timing is extremely critical—nearly impossible in a unix environment—read the comments to the FlexFax package for details).

5.2 Operating Systems

This section describes problems that have occured while porting the package to various operating systems.

If your system is not in the list, that doesn't mean it won't work. It just means that I didn't get a report (or a port) for that system yet.

mgetty+sendfax should work on most unixoid operating systems, as long as they provide `SysV termio` or `POSIX termios` call for tty management. `BSD sgtty` support isn't finished yet.

For best results, your system should have `select(S)` or `poll(S)`, but if both functions are not available or don't work on ttys (`poll(S)` is known to do this on SVR3.1 and earlier), you can use a kludge, `-DUSE_READ`.

Besides that, you'll need some fiddling with the header files to get all defines and prototypes right.

Further, you'll have to check '`tio.c`', function `tio_set_flow_control()`, whether the way hardware flow control is activated works on your system. Most likely, it won't — that's one of the major portability problems. If you change something and get it to work, please send me patches. (You're welcome to do so anyway).

5.2.1 Generic problems and common mistakes

There are a few things that can go wrong on all supported operating systems. This section is meant to give a few hints how to find them.

- `mgetty` is correctly set up in '`/etc/inittab`', but `ps` doesn't list `mgetty` anyway.

  This is usually caused by not notifying the `init` of your changes to '`/etc/inittab`' (or '`/etc/ttytab`' on BSD). You have to signal `init` with a HUP signal (`kill -1 1`) that it should re-read this file.
- `mgetty` is setup correctly, runs a while, then stops being listed by `ps`, and `init` complains something like '`respawning to fast, disabling ...`'.

  This means, there is an error in the mgetty set up. If the `mgetty` process quits too fast (because of an error), and too often in a row, `init` will assume that something is going crazy, and disable the process. You can re-enable it with `kill -1 1`, but you should check the mgetty log file (!!!!!) before, to find out **why** it failed in the first place

5.2.2 SCO Unix 3.2.2 (ODT 1.0 / 1.1)

No major twiddling needs to be done. If your `select()` refuses to sleep less than one second, use poll(S) instead (set `-DUSE_POLL` in the CFLAGS section of '`Makefile`').

Use the modem-control devices for mgetty and dial-outs (e.g. '`/dev/tty2A`'), or (far better), use FAS with minor number of 80+port, using full RTS + CTS handshake, and non-blocking opens ('`/dev/ttyF01`') - the original SCO serial driver is slow, unreliable and doesn't do proper hardware flow control. See Section 5.2.3 [SCO-324], page 46.

Ignore all the warnings about "passing arg 2 of signal from incompatible pointer type". They are caused because the SCO 3.2.2 development system header files are somewhat unusual.

If it doesn't work, or some weird things happen on login (e.g., zmodem downloads do not work), try compiling with -DSYSV_TERMIO. I had some problems with Posix termios on SCO ODT 1.0.

5.2.3 SCO Unix 3.2.4 (ODT 2.0 and ODT 3.0)

I'm using mgetty on SCO 3.2v4(.2) (in fact, developing it there), so be assured: it works.

I consider the way that hardware flow control is handled on SCO to be broken, so I *strongly* recommend using the `FAS` serial driver (version 2.11 or higher, earlier versions may crash the system), to be found on your nearest comp.sources.unix archive. With `fas`, use the devices with a minor number of '`80+port number`' for best results. Make sure that your modem enables the '`DSR`' line, because otherwise, `FAS` won't do hardware handshake.

If you don't use `fas`, I've been told that you have to use the '`modem control`' lines, that is, the "uppercase" lines, e.g. '`/dev/tty1A`', because SCO's serial driver won't do any hardware flow control at all on the "lowercase" lines. Be warned, the driver will also disable hardware flow control if you use Xon/Xoff flow control (no way to use both). Since I do not have a SCO system without `fas`, I'd like to hear very much about results on one.

Also, you've to define `LOCKS_LOWERCASE`, since that's the convention on SCO Unix and most other programs expect it.

If mgetty works only partially, but hangs the moment '`/etc/issue`' is printed, before the 'system!login:' prompt is output, you may have to change the following line of '`mgetty.c`' (around line 780):

```
/* honor carrier now: terminate if modem hangs up prematurely
 */
tio_carrier( &tio, TRUE );
```

to:

```
tio_carrier( &tio, FALSE );
```

But before you do this, make sure that your modem enables the CD line while a carrier is present (Hayes modems: 'AT&C1') and also enables the DSR line (otherwise the port will block once CLOCAL is removed).

This could have been a problem specific to Uwe's dumb AST-compatible fourport card, but I do not think so.

Compilation issues:

Ignore warnings about 'struct utimbuf' and 'struct timeb', they are caused by improper include files. On SCO 3.2v4.2, ignore the warnings about the getopt() prototype, or change prototype or include files.

Installation:

SCO provides two utilities to manipulate '/etc/inittab', enable and disable. Those tools work only if you have specified a gettydefs tag on the mgetty command line, otherwise they will complain about "not a valid tty". So, either append the gettydefs tag (mgetty will ignore it if not compiled with USE_GETTYDEFS) or change '/etc/inittab' manually.

5.2.4 Linux

In current stable kernels (i.e. the latest 2.0.xx release or even one of the 1.2.xx series) and current shared libraries (libc version 5.x) there should be no bug to interfere with this software.

If you have a really old Linux system, notice that Linux kernel versions prior to 0.99pl15 have a bug in the serial handshake code, that is, if the 'CRTSCTS' flag is set, software flow control (XON/XOFF) won't work. The alarm() call is broken in 4.1 and 4.4.2 libraries, which sometimes results in aborting the fax receiving.

On very recent systems using the GNU Libc, you **must** use mgetty 1.1.10 or higher, as the timeout handling on all previous versions doesn't work under glibc. Unfortunately, this means that the version of mgetty shipped with RedHat 5.0 (1.1.9) won't work. Upgrade!

Hardware handshake (RTS+CTS) on Linux works flawlessly (but only if mgetty is compiled with POSIX_TERMIOS, but that is default on Linux anyway). Nevertheless, the scrts.c program in 'contrib/' is still provided, it has some other uses, too.

Linux has no poll(S), so, don't #define USE_POLL, and the default, USE_SELECT, will be used.

Important note: Use the '/dev/ttyS*' devices for getty and for dial-out (that is, for kermit, uucico, cu, seyon, ...) - **never** use '/dev/cua*'. Dialing out on '/dev/cua*' will result in the error message "device busy". (There are reasons why mgetty cannot use the "'ttyS*' vs. 'cua*' kernel locking mechanism", see below). If **all** programs agree on using '/dev/cua*' only, it will work, too - but they have to agree on one variant.

For some background about 'ttyS' vs. 'cua', you might want to read a mail from the author of the Linux serial drivers, Ted Ts'o, posted to the Linux-PPP mailing list. I have included it in 'doc/ttyS-cua.txt'.

Some guys seemingly can't resist posting misinformation to the net all the time, don't believe 'em. The '/dev/cua*' devices are **not** different from the '/dev/ttyS*' devices concerning data flow or modem control lines. The only difference is how the device reacts if you do an open(): Opening '/dev/ttyS*' normally blocks until the "carrier detect" line goes active (unless open() is called with the O_NDELAY flag; mgetty and all dial-out programs do that), and opening '/dev/cua*' will return an error message (errno=EBUSY) if another process has the device already open, thus *preventing dial-out on '/dev/cua*'* if mgetty is active on '/dev/ttyS*'.

We use '/dev/ttyS*' all the time for dial-in *and* for dial-out, and believe me, it works, and it's the *only* combination that will work properly. The kernel locking mechanism only works if you use modem auto-answer (the getty process sleeps until the modem gets a carrier), and mgetty uses manual answer (it waits for the RING message from the modem), which will save your callers a lot of grief because their calls will only be answered if your computer is ready to receive a call. Part of the motivation for writing mgetty was being tired of losing lots of money for useless calls to a hung machine.

I'd recommend against using '/dev/modem' as a link to the real device, but if you do that, make it a **hard link** to the appropriate '/dev/ttyS*'. A soft link will cause problems with the device ownership because of a peculiarity in the linux chown() implementation (that I refuse to work around).

If you get into trouble with write permissions on ttySx, you may add a new group 'dialout' to '/etc/group', then chown .dialout /dev/ttySx your device, and add your users to the dialout group. Don't forget to add the system user 'uucp' to that group (UUCP needs to have modem access), and make sure, port-group in mgetty's configuration file is set up correctly. The concept of such a dialout group is already used in most Linux distributions today.

There are various different init and last programs out there, some work with mgetty, some don't. If you get some strange output from who or last and are using a different init program than the sysv init, try to define -DNO_SYSVINIT. That should help.

I've been told that it's necessary to do that if you use the simple-init.

Anyway, I can only **strongly** recommend to switch over to SysVinit if you use simple init, since the latter seems to be severely broken regarding utmp and controlling tty handling.

If you have problems because of an uninitialized TERM environment variable (which really isn't getty's job, but getty_ps insists on doing it, and people rely on it), use the term ... config file option to set it according to your needs.

If you're experiencing problems with hanging /bin/login processes, See Section 5.3.3 [login-hang], page 55.

Recently, I have received a number of bug reports concerning operation in systems using one or more **Cyclades serial boards**. There is some incompatibility between the standard Cyclades driver and the GNU CC 2.7.2, which can result in system lockups. Upgrade to the very latest Linux driver, which can be found on ftp.cyclades.com. I have received reports that the driver version 2.1 works fine (85 modems on one system, connected to 3 Cyclom Ye boards!).

5.2.5 ISC

The ISC port has been done by Russell Brown, russell@lutton.lls.com. Thanks!

First of all, define `-DISC` in the Makefile. This will take care of some minor problems. Then, link with `-linet -lpt`.

If you have a ISC Unix 4.0, you may have to define `-D_POSIX_SOURCE` to get around some include file problems and link `-lcposix`.

If you have problems with the AWK programs in the '`fax/`' shell scripts, try defining `AWK=nawk` in the '`Makefile`'. That should take care of those problems.

Again, for best results I recommend using the FAS serial driver, and using a port with a minor number of 80+portno (ttyF01 etc.)

If you use a Digiboard smart serial cart (e.g. the digiboard pc/8e), use the '`/dev/ttyi*`' devices instead of '`/dev/cui*`', becaus only the former ones honour carrier drops (If you use '`cui*`', your processes won't die if the modem unexpectedly hangs up)

5.2.6 SVR4 Unix

mgetty has been ported to SVR4 now (many thanks to Bodo Bauer, <bodo@hal.nbg.sub.org>, Martin Husemann, <martin@bi-link.owl.de> and Marc Boucher <marc@cam.org>).

As far as I know, it's sufficient to add '`-DSVR4`' to the CFLAGS in Makefile. If you have any problems or suggestions, please report them also to the people above, since I do not have a SVR4-System to run tests on.

If you use the `SAS` serial driver (streams-version of `FAS`) and want to force `sas` to use hardware-handshake all the time, use a device with a minor device number of `80+port number` (see the `sas` manual for explantations). If you use a port with a minor device number of `7*16+i`, `mgetty` is able to switch hardware handshake on and off according to the flags set in policy.h, using '`sys/termiox.h`'. (Well, it works - but apparently fax reception doesn't work with this minor device number. Symptom: only one byte is received during fax reception (0x00). Anybody any clue?).

If you use `FAS`, use the devices with a minor device number of `80+port number` (as usual).

5.2.7 SVR4.2 - Onsite Unix, UnixWare, ...

Basically, SVR4.2 is quite similar to SVR4, but you have to watch out for some details (defining `-DSVR42` in the Makefile will do it for you).

Most important, the *termiox* interface via the `TCGETX` / `TCSETX` `ioctl()`s does not seem to work any longer - the calls return an error, and the port behaves strangely. If you're experiencing this, please try commenting out the corresponding code in '`tio.c`', funtion `tio_set_flow_control()` and mail me whether that make it work.

Further, using `USE_POLL` or `USE_READ`, won't work. The default of `USE_SELECT` is OK.

To enable hardware handshake, use the tty device with the trailing "h", e.g. '`tty01h`'. On the other one ('`tty01`'), the driver won't do H/W handshake.

Depending on the configuration, parallel dial-out with Taylor-UUCP may fail (uucico complaining that it cannot set `CLOCAL`), in that case, you've to recompile Taylor with different settings for the `termio` selection (POSIX vs. SYSV).

Many thanks to Joerg Weber (joerg@interface-business.de) for finding all those problems.

Ed Hall (edhall@rand.org) found another major glitch on UnixWare 4.2: if you run `sendfax` without setting `ignore-carrier true` in '`sendfax.config`', and `sendfax` switches off carrier detection at the end of the very last page, the kernel code messes up something and bytes get lost. The modem then returns funny error codes, like, for example,

```
Transmission error: +FHNG:44 (Unrecognized Transparent data command)
```

If you're experiencing this, just set `ignore-carrier true` and everything should work just fine (please tell me in any case whether it was necessary, because if it happens for other people as well, I'll make this permanent on SVR42).

See Section 5.2.11 [Solaris2], page 52.

5.2.8 BSD-like flavours of Unix

A port to 386BSD, NetBSD, FreeBSD has been done by Martin Husemann, martin@bi-link.owl.de, and Gunther Shadow, gusw@fub46.zedat.fu-berlin.de.

I think it works quite well, except that the `VTIME` mechanism to timeout `read()` calls doesn't work in older *BSD versions. If `mgetty` hangs, with the last line in the log file being something like "waiting for line to clear", upgrade your kernel, or, if you can't do that, compile `mgetty` with `-DBROKEN_VTIME` (in that case, select() will be used).

For older versions of BSD Unix that do not have '`termios.h`', you'll have to complete the unfinished support for '`sgtty.h`' in '`tio.c`' and '`tio.h`'.

Generally, BSD Unices do not have a '`/etc/inittab`' as system V has. Instead, they have '`/etc/ttys`' (or sometimes '`/etc/ttytab`' on really old BSD systems). Thus, you have to enter a line like

```
cua0    "/usr/sbin/mgetty -x 3 cua0"       vt100
```

or

```
cua00 "/usr/sbin/mgetty -x 3"    vt100 on insecure
```

there. See the corresponding manpage for an exect description of the files format on your system. Don't forget to remove (or comment out) the original getty on the corresponding '`/dev/tty*`' line.

5.2.9 IBM's AIX Operating System

Chris Lewis, Harald Milz and Michael Staats have done excellent work on porting `mgetty` to AIX. Since then, I have taken over and actively use mgetty+sendfax on AIX for customer systems, and everything is very well tested now.

On AIX, many people do not want to manipulate '`/etc/inittab`' directly, instead, use some system administration tools (like 'smit'). To ease `mgetty` installation on AIX, Michael Staats has provided a small shell script, `inittab.aix`, that will help you setup your '`inittab`' after you've run `make install`. Just call it with the name of the tty you're modem is connected to, e.g.

`./inittab.aix tty0`.

I have received a couple of problem reports on AIX 4.1 where "suddenly" the modem line stopped working and all mgetty reported were error messages. If that happens to you, set `toggle-dtr no` in '`mgetty.config`'. AIX 4.1 doesn't seem to like programs that fiddle with the modem control lines.

Christoph Brinck (cb@medat.de) has also reported that it's necessary to enable the "dtropen line discipline" for the serial line you're using (whatever that may mean). This is done with the command:

```
chdev -l 'tty1' -a flow_disp='rts'
chdev -l 'tty1' -a open_disp='dtropen'
```

or via the '`chgtty`' part of `SMIT` (but I think that's the default setting for new ttys anyway).

Hardware and Software flow control work fine on AIX 3.x and AIX 4.x now.

5.2.10 SunOS 4.1.1 and up

mgetty has been ported to SunOS, and seems to work quite well. If you use SunOS, please send me a brief report about your results.

Thanks to Earl Hartwig, earl@fozzi.ocunix.on.ca, for the initial port.

For compilation, please define `-Dsunos4`.

In '`policy.h`', you've to adapt the location of the LOCK files.

In the Makefile, set `ECHO='...'` to `/usr/5bin/echo`, because the standard one doesn't support escape codes like '`\n`' or '`\c`'. Alternatively, if you don't have the System5 options installed, use `mg.echo`.

If a fax reception hangs shortly after the '`+FCON`' is seen, please try setting `FAXREC_FLOW` to Xon/Xoff (`FLOW_SOFT`). Hardware handshake has problems on SunOS versions without the "Jumbo TTY" patch.

If fax sending hangs mysteriously between the first and the second page, you're likely to have a modem that drops DCD during pages. As SunOS' serial drivers are dumb, reception of data will fail if DCD is low and handshake is set to RTS/CTS. So, set `ignore-carrier yes` in '`sendfax.config`' **and** `#define FAXSEND_FLOW FLOW_SOFT` in '`policy.h`' and fax sending should work.

Please use the "outgoing" devices ('`/dev/cua*`') for mgetty and dial-outs, using the "incoming" devices ('`/dev/tty*`') will make dialout impossible. Further, carrier detect (DCD) is only honoured on the '`cua*`' lines.

It is **very** strongly recommended that you install the "jumbo tty patch" (patch number 100513-05 for 4.1.2 or .3, patch number 101621-01 for 4.1.3_u1 and up) because it will fix a lot misbehaviour of the serial line drivers.

Please read also the generic BSD section.

5.2.11 Solaris 2.3 and up

mgetty runs successfully and without trouble under Solaris 2.3, 2.4 and 2.5.1 (later versions should also work, but I didn't get any report so far). For compilation use `-Dsolaris2`. With Solaris 2.3 it's recommended to use GNU gcc, but with Solaris 2.4 it compiles fine with the SPARCompiler C 3.0.1. Define `CC=cc`.

In '`policy.h`' you have to define '`term/a`' or '`term/b`' as `FAX_MODEM_TTYS`. Don't use the outgoing devices '`/dev/cua/*`'!!

As `DEVICE_GROUP` you should configure '`uucp`'. If you want allow to normal users to dial out, add all users allowed to do that, to the group '`uucp`'. Then it's important to change the permissions of '`/var/spool/locks`' from the default permissions '`drwxr-xr-x`' to '`drwxrwsr-x`'. Make sure that it's owned by '`uucp.uucp`'. Otherwise no one wanting to dial out is able to create a lock file. The `FILE_MODE` in '`policy.h`' must be '`0660`' as well.

If you don't want allow anyone to dial out you should set `FILE_MODE` to '`0640`' or '`0600`'.

For the `"notify mail"` message to look best, define `MAILER` in '`policy.h`' to */usr/lib/sendmail* and define `NEED_MAIL_HEADERS`. So a proper subject header is created. Nevertheless, `/usr/bin/mailx` works, as it is the default for SVR4.

If everything compiled well and you did `make install`, you have to add an entry to '`/etc/inittab`' like the following:

```
ya:234:respawn:/usr/local/sbin/mgetty -s 38400 -x 3 term/a
```

Don't use the Solaris `admintool` *to create any port monitoring entries in* '`/etc/saf/_sactab`' *and* '`/etc/saf/zsmon/_pmtab`'*.*

Many thanks to Stefan Muehlenweg (muehlenw@samhh.hanse.DE) for this section.

Solaris (as all Sun operating systems) seems to be somewhat weird concerning its handling of the RTS line. I have received two reports that 'sometimes' a modem hangs during initialization, and won't talk to mgetty anymore. In these cases, the problem went away when the modem (an USR Sportster) was set to `AT&R1`, which means 'ignore RTS line'. Thanks to Valerio Di Giampietro for detailing this.

5.2.12 AT&T 3b1

Glenn Thobe and Chris Lewis have ported mgetty+sendfax to AT&T's 3B1 machines, it should compile without changes to the source (but define -D_3B1_ in the Makefile).

These ports are to two different environments:

Glenn's port was with GCC (ANSI C) and an add-on select() library routines. Chris's port was with stock 3b1 C, without select(). Both seem reliable.

Some further hints concerning a select() library and the pbmplus tools can be found in the file '`contrib/3b1`' which are the notes from Glenn's port.

Chris's port relies simply on the suggested definitions (especially `-DUSE_READ`) in the Makefile. Chris suggests that you use select() if you already have it for some other reason, but that it seems to work just fine without it.

Right now, I think mgetty won't compile with the standard C compiler (it will with gcc), because the stuff I do in the '`conf_*.c`' source files is a little bit too hard for it. I'll work on it.

5.2.13 The HP-UX operating system

`mgetty` runs on HP-UX, but that's very much all I can say about it (I don't know anything about HPs, except that they are somewhat strange).

Currently, documentation for HP-UX is a bit lacking - if you know something about `mgetty` on HP-UX, please contribute.

5.2.14 The NEXTSTEP operating system

NEXTSTEP is lacking quite a few commands used by mgetty and its tools: Luckily, they are available from the GNU project (please mail me if I forgot any), e.g. gawk, cut (from textutils), id and logname (from sh-utils). All of them can be compiled for NEXTSTEP without problems, and it's certainly a good idea to install them anyway. ghostscript needs some fiddling for NEXTSTEP, but it's available precompiled on the arcives as GSPrintFilter. If you don't intend to use faxspool and friends, you may succeed without installing the forementioned utilities.

Having said this, there are two different ways to compile `mgetty` for NEXTSTEP, each with their pros and contras:

- With termios interface: To use `mgetty`'s termios interface (the default), you have to link against NEXTSTEP's buggy POSIX library.

  In the Makefile, add "-posix -DBSD" to CFLAGS and LDFLAGS.

  Drawbacks: The log file will be corrupted due to an append bug in libposix.

  Use this only if you want to try vgetty. In all other cases it's probably better to choose the second variant:

- With sgtty interface: Ben Stuyts '`benst@stuyts.nl`' has done an effort to make a `mgetty` port using the BSD sgtty style interface to the serial ports. This port doesn't need the POSIX library.

  In the Makefile, add "-DNEXTSGTTY -DBSD".

  This port has been tested with m68k and i386 machines.

  For i386 machines, you may want to use the /bin/login replacement '`modem-login`' in `mgetty`'s contrib directory. NEXTSTEP's login has problems with 8-bit logins. I've got one report from Ben Stuyts, though, that '`modem-login`' does fix problems on "Black NeXT"'s as well. Just try it.

  Drawback: You can't compile vgetty with sgtty support yet! If you want to use vgetty with NEXTSTEP, you'll have to stick with the termios port. There are problems with the LNOHANG bit not being acknowledged correctly, therefore hanging the modem when the caller hangs up.

For i386 machines, it's wise to use the /dev/cudfX (hardware flow control,...) devices. I'm using them with NeXT's most recent serial port driver, Mark Salyzyn's Mux driver also supports them.

For m68k machines, you have to stick with /dev/cufa.

This are the settings I use in the Makefile:

```
CC=cc
CFLAGS=-DNEXTSGTTY -DBSD -O2  (-posix -DBSD for termios port)
INSTALL=install -c -o root -g wheel
spool=/usr/spool
SBINDIR=$(prefix)/etc
ECHO="mg.echo"
INSTALL_MECHO=mg.echo
AWK=gawk
```

Furthermore, you should define binary lock files, in policy.h:

```
#define LOCKS_BINARY 1
```

Finally, when using NEXTSTEP's cc, you need to run "make noident" in the first place to remove the #ident directives from the source files.

If you have questions, comments or suggestions regarding `mgetty` with NEXTSTEP, feel free to contact Gregor Hoffleit '`flight@mathi.uni-heidelberg.de`' or Ben Stuyts '`benst@stuyts.nl`'.

- Coming real soon now:

  How to use `mgetty` to send faxes via the NEXTSTEP fax panel

  (using Kevin Peckover's ps2g3 package, available as '`ftp://peanuts.leo.org/pub/next/Tools/postscript/ps2g3.s.tar.gz`').
- Coming soon:

  How to patch and use `mgetty` to spool received faxes in the NEXTSTEP fax system.

If I understand this correctly, "Darwin" is the underlying operating system that Apple MacOS X is based upon.

Edwin C Wirth <ecwirth@mac.com> gave me an account to his "Darwin Kernel Version 5.1" machine (running MacOS 10.1.1 on top of it) so I could port `mgetty+sendfax` to that OS. Thanks to that, Darwin is supported from 1.1.28 on.

Compilation is pretty straightforward: use the default CFLAGS and LIBS settings in the Makefile, set `CC=cc`, and then run `make`. It will spit out a ton of warnings about '`undefined or invalid # directive`' - but those are harmless, it's just cpp not understanding `#ident`.

At run time, as far as I could see, the internal Mac modem is available on '`/dev/tty.modem`'. The macintosh I had for testing also had "KeyUSA" USB-to-Serial ports that appeared as '`/dev/tty.KeyUSA28X913.1`' and '`.2`' - so other serial ports should also appear as '`/dev/tty.*`' devices. Note: there seem to be "cu" devices as well ('`/dev/cu.modem`'), as in older Linux variants. I did only try sending faxes via '`tty.modem`' yet, but to resolve locking issues, it might be neccesary to use '`cu.*`' devices. To be continued...

Mgetty should be run from '`/etc/ttys`' as in other BSDs, see `man ttys` (there is no '`/etc/inittab`').

Open issues:

is it better to use '`/dev/cu.*`' or '`/dev/tty.*`'?

how is tty locking done? Couldn't find UUCP lock files

any tricky things about logging in remotely? utmp/wtmp?

where should the log files go to? /var/log doesn't seem to exist.

5.3 General problems

5.3.1 pbmtog3

The pbmtog3 program from the 'pbmplus' distribution produces G3 data that does not adhere to the T.4 standard. The initial EOL code is missing, and the lines are not always 1728 pixels wide. So, some fax machines won't accept the output at all (not printing even one line), and others will complain.

A fix for this problem is available: I have included a patch for pbmtog3, called 'patches/pbmtog3.p1', that will fix the problems. (Oh, by the way, if you try to send a fax generated with an unpatched pbmtog3, sendfax will complain that it doesn't like the file . . . I've added a small sanity check to spare me the time browsing through the logfiles guessing *why* sendfax failed (won't work if the file has been processed by g3cat, though!)). If you use the pbm2g3 program that is shipped with mgetty, there is **no need to patch anything**.

Basically, there is no need at all to use 'pbmplus'' pbmtog3 program any more, since mgetty includes an own copy. I just wanted to warn you.

Anyway, my program is lots faster . . . .

5.3.2 Lock files

Kermit et.al. cannot dial out while mgetty is running (modem responses are eaten by mgetty)—what's wrong?

Most propably, you have not configured the LOCK and LOCKS_BINARY defines in 'policy.h' properly. Make sure that the lockfiles kermit (or cu, pcomm, seyon,. . .) expect are in the path specified in LOCK and set LOCKS_BINARY to 1 if they do not write the PID of the locking process in ascii (10 bytes) to the lock file but as a 4-byte integer instead. Mgetty and Sendfax will understand both types of lock files, but if LOCKS_BINARY is not set properly, other programs may not understand the lock file.

Also, make sure that both processes use the same name for the device. (i.e., mgetty locking '/dev/ttyS0' and kermit locking '/dev/modem' will definitely fail.)

5.3.3 mgetty works, /bin/login hangs

A problem seen fairly often on directly connected serial lines (mgetty -r), and seldomly on modem lines, is that mgetty works flawlessly, but /bin/login just hangs instead of prompting for the user password.

The reason for this is that many login programs reopen '/dev/tty' (the controlling tty of a process) to make sure they have full control over the password entered by the user (for example, to prevent snooping). This will block on some systems if the DCD (carrier detect) line coming from the modem or the other machine is low. Notably those systems are those that have callin/callout device pairs for one serial device, e.g. Linux, SunOS, SCO/FAS, etc.

The fix is easy: make sure that the DCD line is high.

If you use a modem, the command to do this is usually `AT&C1` (but check with your modem manual).

If you're using a direct null-modem connection to another host, the recommended solution is to wire DCD on your side to the DTR line on the other side and vice versa. That way, when the remote machine "hangs up" (calling program exits and DTR drops), your host will get notified as well. This is what a properly wired (!) "null-modem cable" does anyway.

If you don't have free lines in your serial cable (classic three-wire approach), wire DCD to the DTR line on your own host, and make sure that mgetty won't toggle DTR upon startup (causing a hangup signal!), e.g. by setting `toggle-dtr no` in '`mgetty.config`'.

5.3.4 ECU 3.20 or earlier on SCO collides with mgetty

ECU releases 3.20 and earlier had a severe bug in the utmp handling that prevented dialing out on a port that mgetty uses. It has been fixed in ECU 3.27. If you run into that problem, please get a newer release. Alternatively, you can use the patch that Uwe Fuerst provided, it can be found in '`patches/ecu320.p1`'.

5.4 Sample Log files

Both mgetty and sendfax can provide logfiles that can be very helpful for debugging and accounting purposes. The amount that is logged is controlled with the default set in '`policy.h`' and the command line argument `-x <level>`. Higher numbers give more details.

At this place, I want to show you some typical cases, so you can compare your log files to those given here and check what is different.

(Note: naturally all the modem initializations, and also some of the modem responses, vary between modem brands!)

All the mgetty log files have been done with log level `L_MESG`, that is, `-x 4`. The sendfax log file was done with `L_NOISE`, `-x 5`.

5.4.1 mgetty, incoming data call

This is a log file of a typical data connection, ZyXEL-to-ZyXEL modems, connect with 19200 bps on a port speed of 38400, login as `"Uartinet"`, the login program called is `/usr/lib/uucp/uucico` (controlled by '`LOGIN_CFG_FILE`', which directs all login names starting with `"U"` to uucico)

```
03/03 22:40:15  check for lockfiles
03/03 22:40:15  locking the line
03/03 22:40:16  lowering DTR to reset Modem
03/03 22:40:16  send: \d\d\d+++\d\d\d[0d]\dATQ0V1H0[0d]
03/03 22:40:20  waiting for ``OK'' ** found **
03/03 22:40:20  send: ATS0=0Q0&D3&H3&N0&K4[0d]
03/03 22:40:20  waiting for ``OK'' ** found **
03/03 22:40:20  send: AT+FCLASS=0[0d]
```

```
03/03 22:40:20  waiting for ‘‘OK’’ ** found **
03/03 22:40:20  send: AT+FAA=1;+FBOR=0;+FCR=1[0d]
03/03 22:40:20  waiting for ‘‘OK’’ ** found **
03/03 22:40:20  send: AT+FLID="49 89 3243328"[0d]
03/03 22:40:20  waiting for ‘‘OK’’ ** found **
03/03 22:40:20  send: AT+FDCC=1,5,0,2,0,0,0[0d]
03/03 22:40:20  waiting for ‘‘OK’’ ** found **
03/03 22:40:20  fax_command: send ’AT+FLPL=1’
03/03 22:40:20  fax_wait_for(OK)** found **
03/03 22:40:21  waiting...
03/03 22:41:28  waiting for ‘‘RING’’ ** found **
03/03 22:41:28  send: ATA[0d]
03/03 22:41:28  waiting for ‘‘CONNECT’’ ** found **
03/03 22:41:42  send:
03/03 22:41:42  waiting for ‘‘
’’ ** found **
03/03 22:41:43 ##### data dev=tty4c, pid=6470, caller=none, conn=’38400/ZyX  16800/V42b’, name=’’, cmd=’/usr/lib/uucp/uucico’, user=’Uartinet’
```

5.4.2 mgetty, incoming fax call

This is a log file of a fax call I got today. It was a fax call with 14400 bps (actually, the call came from another faxmodem, but you won’t see that in the log file), using high resolution. One page was received, the connection time was 33 seconds. No errors occured.

(All the stuff up to "waiting..." is identical to the example above, so I don’t list it again)

```
[...]
03/03 21:39:32  waiting...
03/03 21:46:22  waiting for ‘‘RING’’ ** found **
03/03 21:46:22  send: ATA[0d]
03/03 21:46:22  waiting for ‘‘CONNECT’’
03/03 21:46:32  found action string: ‘‘+FCON’’
03/03 21:46:32  action is A_FAX, start fax receiver...
03/03 21:46:32  fax_wait_for(OK)
03/03 21:46:36  fax_id: ’+FTSI:+31 20 6147110       ’
03/03 21:46:36  transmission par.: ’+FDCS:1,5,0,2,0,0,0,0’** found **
03/03 21:46:36  fax_command: send ’AT+FDR’
03/03 21:46:36  fax_wait_for(CONNECT)
03/03 21:46:37  fax_id: ’+FTSI:+31 20 6147110       ’
03/03 21:46:37  transmission par.: ’+FDCS:1,5,0,2,0,0,0,0’** found **
03/03 21:46:38  fax_get_page_data: receiving /usr/spool/fax/incoming/ffd764c9e4d-+31-20-6147110.01...
03/03 21:46:51  fax_get_page_data: page end, bytes received: 24933
03/03 21:46:51  fax_wait_for(OK)
03/03 21:46:51  page status: +FPTS:1** found **
03/03 21:46:53  fax_wait_for(CONNECT)
03/03 21:46:55  connection hangup: ’+FHNG:000’** found **
03/03 21:46:56 ##### fax dev=tty4d, pid=4807, caller=none, name=’’, id=’+31 20 6147110       ’, +FHNG=000, pages=1, time=00:00:33
```

5.4.3 mgetty, logging into syslog

If your system has a `syslogd` and the `syslog()` C function, mgetty can send parts of its log files to the '`syslog`' (For details, see comments in '`policy.h`'). Not all the information from the log file is logged here (to avoid clobbering the syslog), just errors and so-called "audit" messages (seen in the log file as lines with "####" at the beginning). These have a fixed format, and could easily be parsed by a program. Let me list a few, and then comment them.

```
Mar  3 18:36:16 greenie mgetty[673]: failed A_FAIL dev=tty4d, pid=673, caller=none, conn='', name=''
Mar  3 18:41:56 greenie mgetty[1866]: fax dev=tty4d, pid=1866, caller=none, name='', id='49 89 3243328        ', +FHNG=100, pages=1, time=00:00:29
Mar  3 21:46:56 greenie mgetty[4807]: fax dev=tty4d, pid=4807, caller=none, name='', id='+31 20 6147110       ', +FHNG=000, pages=1, time=00:00:33
Mar  3 20:45:59 greenie mgetty[4038]: data dev=tty4d, pid=4038, caller=none, conn='38400/V.32  9600/MNP5', name='', cmd='/bin/login', user='mbox'
Mar  3 22:41:43 greenie mgetty[6470]: data dev=tty4c, pid=6470, caller=none, conn='38400/ZyX  16800/V42b', name='', cmd='/usr/lib/uucp/uucico', user='Uartinet'
```

Those five lines are one failed call, two fax calls, one of them failed and one successful, and two data calls, one of a human caller, loggin into the BBS system, and one of a calling uucico.

It looks very confusing until you understand the system behind it. The first fields specify date and time, originating host (greenie is my machine) and program (mgetty). The next field specifies the type of the connection made: '`fax`', '`data`' or '`failed`' - the latter one usually means failure to initialize the modem or failure to connect to a calling modem, resulting in the well-known '`NO CARRIER`' message...

The '`dev`' and '`pid`' fields specify the line and process ID of the mgetty process writing that line.

The '`caller`' and '`name`' fields give caller ID information - if none is available (as it is here in Germany), or if your modem doesn't handle it, it will list '`none`' and '`''`', respectively.

For fax calls, additional informations given are the remote station ID ('`id='...'`'), the hangup code ('`+FHNG=...`', 0 means "ok"), the number of pages and the connection time.

For data calls, the string that the modem returned after '`CONNECT`' is listed as '`conn='...'`'. The string that was entered at the login prompt is listed as '`user='...'`', and the program that is called to do the login is given as '`cmd='...'`'. Usually this will be `/bin/login` unless you have some special system setup for fido or uucp callers - as I have here, as you can see above.

5.4.4 sendfax, sending a single page

This is a simple one-page fax that I sent some days ago. Just a single page, f1.g3, to the phone number 2710834. No errors of any kind occured.

```
02/18 11:10:05  sending fax to 2710834
02/18 11:10:06  checking f1.g3
02/18 11:10:06  makelock(tty4c) called
02/18 11:10:06  do_makelock: lock='/usr/spool/uucp/LCK..tty4c'
02/18 11:10:06  lock made
02/18 11:10:06  fax_open_device succeeded, tty4c -> 4
02/18 11:10:06  fax_command: send 'AT'
02/18 11:10:06  fax_wait_for(OK)
```

```
02/18 11:10:06  fax_wait_for: string 'AT'
02/18 11:10:06  fax_wait_for: string 'OK'** found **
02/18 11:10:06  fax_command: send 'AT+FCLASS=2'
02/18 11:10:06  fax_wait_for(OK)
02/18 11:10:06  fax_wait_for: string 'AT+FCLASS=2'
02/18 11:10:06  fax_wait_for: string 'OK'** found **
02/18 11:10:06  fax_command: send 'AT+FLID="49 89 3243328"'
02/18 11:10:06  fax_wait_for(OK)
02/18 11:10:06  fax_wait_for: string 'AT+FLID="49 89 3243328"'
02/18 11:10:06  fax_wait_for: string 'OK'** found **
02/18 11:10:06  fax_command: send 'ATL7M1'
02/18 11:10:06  fax_wait_for(OK)
02/18 11:10:06  fax_wait_for: string 'ATL7M1'
02/18 11:10:06  fax_wait_for: string 'OK'** found **
02/18 11:10:06  fax_command: send 'AT+FDCC=1,5,0,2,0,0,0,0'
02/18 11:10:06  fax_wait_for(OK)
02/18 11:10:06  fax_wait_for: string 'AT+FDCC=1,5,0,2,0,0,0,0'
02/18 11:10:06  fax_wait_for: string 'OK'** found **
02/18 11:10:06  fax_command: send 'AT+FBOR=0'
02/18 11:10:06  fax_wait_for(OK)
02/18 11:10:06  fax_wait_for: string 'AT+FBOR=0'
02/18 11:10:06  fax_wait_for: string 'OK'** found **
02/18 11:10:06  fax_command: send 'AT&H3D2710834'
02/18 11:10:06  fax_wait_for(OK)
02/18 11:10:07  fax_wait_for: string 'AT&H3D2710834'
02/18 11:10:39  fax_wait_for: string '+FCON'
02/18 11:10:39  fax_wait_for: string '+FNSF:00 00 00 00 '
02/18 11:10:39  fax_wait_for: string '+FCSI:       49 89 2710834 '
02/18 11:10:39  fax_id: '+FCSI:       49 89 2710834 '
02/18 11:10:39  fax_wait_for: string '+FDIS:1,3,0,2,0,0,0,4'
02/18 11:10:39  fax_wait_for: string 'OK'** found **
02/18 11:10:39  fax_send_page("f1.g3") started...
02/18 11:10:39  fax_command: send 'AT+FDT'
02/18 11:10:39  fax_wait_for(CONNECT)
02/18 11:10:39  fax_wait_for: string 'AT+FDT'
02/18 11:10:45  fax_wait_for: string '+FDCS:1,3,0,2,0,0,0,4'
02/18 11:10:45  transmission par.: '+FDCS:1,3,0,2,0,0,0,4'
02/18 11:10:45  fax_wait_for: string 'CONNECT'** found **
02/18 11:10:45  waiting for XON, got:[0a][11]
02/18 11:10:45  sending f1.g3...
02/18 11:11:03  sending DLE ETX...
02/18 11:11:03  fax_wait_for(OK)
02/18 11:11:16  fax_wait_for: string 'OK'** found **
02/18 11:11:16  fax_command: send 'AT+FET=2'
02/18 11:11:16  fax_wait_for(OK)
02/18 11:11:16  fax_wait_for: string 'AT+FET=2'
02/18 11:11:25  fax_wait_for: string '+FPTS:1'
```

```
02/18 11:11:25  page status: +FPTS:1
02/18 11:11:26  fax_wait_for: string '+FHNG:00'
02/18 11:11:26  connection hangup: '+FHNG:00'
02/18 11:11:26  (Normal and proper end of connection)
02/18 11:11:26  fax_wait_for: string 'OK'** found **
02/18 11:11:26  fax_send: 'AT+FCLASS=0^M'
02/18 11:11:26  removing lock file
```

5.5 How to get the mentioned software by FTP?

Most of the software mentioned in this document should be available on most major ftp sites. Nevertheless, I've got so many questions about the software that I'll list some ftp sites here.

Furthermore, I'll list some other software that may be interesting if you plan to use `mgetty+sendfax` in different environments than I do.

- `mgetty+sendfax`

  The current release can usually be found at:

  sunsite.unc.edu:'`/pub/Linux/system/serial/getty/mgetty+sendfax*`'

  tsx-11.mit.edu:'`/pub/linux/sources/sbin/mgetty+sendfax*`'

  alpha.greenie.net:'`/pub/mgetty/source/*`'

  ftp.leo.org:'`/pub/comp/os/unix/networking/mgetty/*`'

  ftp.mathematik.th-darmstadt.de:'`/pub/linux/mirrors/misc/mgetty/*`'

  linux.nrao.edu:'`/pub/src/mgetty/*`'

  ftp.chem.tue.nl:'`/pub/mgetty/*`'

  http://linux.mit-lab.co.kr/mgetty/*

  ftp.uvt.ro:'`/pub/mgetty/*`'

  ... and on all sites that mirror SunSITE.unc.edu or tsx-11.

  The most recent beta release is always placed in the listed directories on alpha.greenie.net, ftp.leo.org and their mirrors on th-darmstadt.de and linux.nrao.edu.

  On SunSITE and TSX-11, you'll only find "stable" releases (even version numbers, no date stamp in the file name).

  Use betas on your own risk!

- `pbmplus`

  I found the pbmplus package (bitmap manipulation tools) at the following places (most other sites mirroring X11 should also have them):

  src.doc.ic.ak.uk:'`/computing/graphics/pbmplus10dec91.tar.Z`'

  wuarchive.wustl.edu:'`/packages/X11R5/contrib-pub/pbmplus10dec91.tar.Z`'

  ftp.germany.eu.net:'`/X11/contrib/pbmplus10dec91.tar.Z`'

  ftp.leo.org:'`/pub/comp/os/unix/networking/mgetty/`'

  The `pbmplus` package has been superceded by the `NetPBM` package which has some more conversion tools (but also some more bugs). It should be available on the same sites. You could also check:

wuarchive.wstl.edu:'/graphics/graphics/packages/NetPBM'

ftp.rahul.net:'/pub/davidsen/source'

ftp.informatik.uni-oldenburg.de:'/???'

The pbmtodot program mentioned in the "fax" chapter can be found, if nowhere else, in the 'mgetty' directory on ftp.leo.org (see above).

- FAS

  Some sites carrying the fas serial driver for SCO Unix, ISC, . . . :

  ftp.fu-berlin.de:'/pub/unix/driver/fas/fas-2.11.tar.gz'

  ftp.germany.eu.net:'/pub/newsarchive/comp.sources.unix/volume27/fas-2.11.0/*'

  src.doc.ic.ac.uk:'/usenet/comp.sources.unix/volume27/fas-2.11.0/*'

  As far as I know, the current version is 2.12, but you should find 2.12 at the same places as 2.11. I never upgraded, 2.11 works very fine for me :)

- GhostScript

  The GNU GhostScript postscript interpreter can be found at all sites carrying GNU Software. The "main" GNU site is *prep.ai.mit.edu*.

  The new version 3.x isn't GNU software anymore, but uses a special license. You can find it in '/pub/ghost/aladdin' on *ftp.cs.wisc.edu*, and on various other mirror sites (e.g. ftp.leo.org).

- hp2pbm

  GhostScript is huge and slow... if you don't need it for other purposes, you could use Chris Lewis' hp2pbm package, placed at

  ftp.leo.org as 'hp2pbm*' (in the mgetty directory mentioned above).

  This package contains all you need to convert ASCII or LJ PCL4 -> PBM, G3, PS, X windows, Epson 24 pin. Coupled with mgetty's g32pbm, it's all you need to send ASCII or HP LJ via FAX, and display/print FAXes on X, PS, LJ, Epson, or via PBM utilities

  It performs better than most of the equivalent PBMPLUS utilities. Indeed, if your application can generate both LJ and PS, generating LJ and converting to G3 via hp2pbm is *much* faster than generating PS and converting via GhostScript.

  [Citing Chris: Incidentally, "g32pbm" plus my "pbm2lj" is much faster than g3tolj. Am not entirely sure why yet. May have something to do with scaling, which g32pbm and pbm2lj don't do.]

- psroff

  ftp.uunet.ca:'/distrib/chris_lewis/psroff3.0pl17/*'

  [Chris: It should be ...psroff/*, but they never answer when I ask them to fix my symlink]

  Contains all you need to generate PS, LJ, or ditroff from ditroff or CAT troff or groff. Together with hp2pbm, you can generate FAXes. Contains quite a number of LJ fonts, font manipulation, and font scaling utilities.

  With the last two tools, you can send, receive, and print FAXes using _just_ mgetty and hp2pbm (and a Laserjet printer ;-), without needing pbmplus or GhostScript or anything else.

- `Ifmail`

  Ifmail is Eugene Crosser's FidoNet (tm) compatible mailer + gateway software for Unix and Linux. Together with `mgetty`, it provides the perfect modem solution, allowing incoming Fido calls as well as Unix-Login and fax. It can be found on:

  tsx-11.mit.edu:'`/pub/linux/sources/usr.bin/ifmail*`'

  sunsite.unc.edu:'`/pub/Linux/system/Fido/ifmail*`'

- `dialog`

  Dialog is a very nice shell tool to simplify input/output functions in shell scripts. All the programs in '`mgetty/frontends/dialog/`' rely on it. It runs on most operating systems (except FreeBSD), as long as they have a `curses` library. The source can be found on

  sunsite.unc.edu:'`/pub/Linux/utils/shell/dialog*`'

- `ps2g3`

  is a small package to integrate `mgetty+sendfax` into the NeXTStep operating system. It can be found on ftp.leo.org in the directory '`/pub/comp/platforms/next/Tools/postscript/ps2g3.s.tar.gz`'.

- `dvips5.47`

  Dvips 5.x and up create a strange kind of postscript that, when converted with Ghostscript 3.x, produces Fax files that can not be faxed to paper fax machines (modems work fine) because the line width isn't exactly 1728 pixels. Dvips 5.47 doesn't do this. It can be found on:

  '`ftp.leo.org:/pub/comp/os/unix/networking/mgetty/dvips*`'

  '`midway.uchicago.edu:pub/tex/dvips/dvips547.tar.Z`'

- `g3vga`

  Is a nice, fast G3 viewer for Linux and SVGALIB (standard console mode, no X11). You can find it in '`.../apps/graphics/viewers`' on all mirrors of SunSITE.unc.edu.

5.6 How to get the mentioned software by UUCP

For those of you that do not have FTP access: all the software mentioned in the last section can also be found on the following UUCP sites:

- GREENIE (my system): ++49-89-35663089, V32bis/V.34, get greenie!~/green.files.gz
- CameloT (Frank Bartels): ++49-89-8948040, V34+, get /pub/ls-lR.gz

6 Thanks

Many thanks to (in no special order):

- Peter Bechtold, peter@fns.greenie.muc.de, for sending me dozens of faxes to test mgetty, for calling me back numerous times after failed attempts to send him a fax with sendfax, ...

  Further, for the idea to use the remote fax id as part of the filename on received faxes.
- Klaus Weidner, klaus@greenie.muc.de, for the original linux port, testing dozens of pre-releases, writing the original texinfo documentation, and finally for writing `vgetty`.
- Lawrence 'dreamer' Chen, lawrence@combdyn.com, for the initial ISC port, and for testing the package with a SupraFAX-Modem.
- Kay Schulz, kschulz@gold.t-informatik.ba-stuttgart.de, for testing on ISC — and telling me that it's possible to ask dozens of questions without having ever read the README file ...
- Georg Edelmann, georg@alpha.saar.de, for testing on Linux, and finding some stupid bugs.
- Uwe S. Fuerst, uwe@phiger.com, for testing on SCO 3.2v4 (and helping me a lot nailing down the problem with dial-in/dial-out)
- Bodo Bauer, bodo@hal.nbg.sub.org, for porting mgetty to SVR4 (though he did quite confuse me by insisting that the fax receiver does not work ...), and later, bodo@suse.de, for his `faxrunqd` daemon.
- Christoph Adomeit, for bugging me long enough to implement XON / XOFF flow control in fax sending / receiving, and for lending me one of his GVC modems long enough to make faxing (well, fax sending) work with it.
- Christopher M. Ward, for testing on SCO with another GVC modem.
- Ralf Stephan, for finding a problem in sendfax whith some modems that lower CD too soon.
- John C. Peterson, for correcting a similar problem in mgetty / faxrec.
- Chel van Gennip, for the `pbmscale`, `g3toxwd` and `g3tolj` programs.
- Glenn Thobe and Chris Lewis, for doing the 3B1 port(s).
- Chris Lewis, for doing all the '`/etc/gettdefs`' stuff, CallerID, space limiting, making the source look really awful (K&R C support), miscellaneous minor fixes, and tolerating my sometimes very unfriendly reactions to his suggestions.
- Caz Yokoyama, for his suggestions concerning faxspool and the mail-to-fax gateway
- Martin Husemann, for SVR4 testing (damn ESIX) and the NetBSD 386 port
- Michael A. Meiszl, mam@mamunx.werries.de, because he asked me to (*grin*) - and because he found `+` changed lots of small, nevertheless annoying things.
- Brent Mosbrook from ZyXEL USA, brentm@zyxel.com, who has been **very** helpful solving some ZyXEL-specific and generic fax questions.
- ELSA Computer GmbH, Germany, for giving me a number of test modems and being pretty suportive overall concering questions of modem firmware, protocol handling, and so on. A big thanks!

- Sam Leffler, sam@sgi.com, for many interesting discussions and insights.
- Christian Starkjohann, cs@ds1.kph.tuwien.ac.at, for important parts of the NeXT port.
- Geoffrey Collyer and Henry Spencer, for writing the `newslock.c` program I use in `faxrunq`.
- Simone "Neko" Demmel, for going to bed early, giving me time to proof read this manual and correct all the nasty bugs.
- Russel Nelson, nelson@crynwr.com, for hosting the mgetty mailing list since December '96.
- Medat Computer GmbH, Munich, for using mgetty+sendfax and paying me for improvements (faxrunqd rewrite and lots of detail work).
- SpaceNet GmbH, Munich, for sponsoring IP connectivity for alpha.greenie.net. Should you ever need an Internet Service Provider in Germany, look at http://www.space.net/...
- Wiebke Baars, for being a really good friend, and for a number of wonderful days spent together.
- . . . and to all others who contributed in some way.

Table of Contents