Httpproxy

Matthias Hopf

COLLABORATORS			
	TITLE : Httpproxy		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Matthias Hopf	July 10, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Httpproxy

1.1 HttpProxy - Contents

```
                ---------------------------------------------------------------------------

      HttpProxy 0.14 public beta
           Date of release: 20. Aug 1996

         A caching proxy supporting the http protocol
For usage with the AmiTCP/Miami or the INet/AS225/Surfer network protocols
         Copyright © 1995-96
                Matthias Hopf
                , EMailware
         All Rights Reserved.

---------------------------------------------------------------------------
```

1.2 HttpProxy - Introduction

```
                Introduction
============

You have AmiTCP V2.2 or greater or  the  INet  /  AS225  /  Surfer  network
package installed and have access to a WWW-server?

You use AMosaic, IBrowse, Weblink (not right now ;-) Lynx,  AWeb  or  Emacs
with  WWW  extension  quiet  often or at least sometimes, maybe even over a
serial line?

You  are  annoyed  of  the  long loading times of new pages or, even worse,
inline graphics?

You don’t like ARexx-Scripts like WWWCache.rexx or the like (I must  admit,
I  never  tried  them  :-]  )  and  you  dislike the caches included in Web
browsers, because

1) you don’t want the html pages to be modified,
2) they don’t work for all links (maybe ?!),
3) you would like to have the pages updated in regular intervals or
4) you can’t use them, because you don’t use AMosaic?
5) you want an automatic solution?
6) you think those Web browser caches are not stable at all?


If  any  answer is no, you may save time and stop reading this, because you
won’t need HttpProxy.


HttpProxy is a caching
                proxy
                 that has several basic configurations.
These are explained in the
                features
                 section.

With the use of HttpProxy you will be able to get any page you have already
visited  without  any  network traffic. That’s much faster than polling the
URLs from the net, of course...
```

1.3 HttpProxy - What is a proxy at all?

```
                Beginners Guide
===============

I assume that you have some very basic knowledge on networking, e.  g.  you
managed  to  get  AmiTCP or INet working on your system, and you understand
terms like ’dial in’.

You used a WWW-browser like AMosaic before and know that the pages you  are
viewing do not reside on your  own  computer  but  are  fetched  from  some
specific  servers  all  around  the  world.  You  already  experienced that
waiting times are really annoying sometimes...
```

Now how about caching all pages you already viewed? Inline graphics for instance don't change very often, so it is absolutely useless to fetch them again and again from the remote server. The same happens to text files, but you won't recognize it that much there (they are usually much shorter).

That's the job of a proxy. A WWW-broswer can be configured to send all URL requests to one specific server instead of sending it to the server that is stated in the URL.

For instance the document for the URL

http://wwwcip.informatik.uni-erlangen.de/user/mshopf/

is fetched from the server "wwwcip.informatik.uni-erlangen.de" when you don't specify any proxy. This could take some time when you live in America or even Australia...

Whenever a request is sent to the proxy, it checks whether it has already accessed the page and the cache entry is valid. In this case, the cache entry is sent to the browser and no network traffic occures.

When there is no valid cache entry for the requested URL, the proxy contacts the server stated in the URL and fetches the document. The document is now cached and future requests to the same URL will make use of the cache.

Most internet providers do have a proxy running, too, it is wise to use it, though you can cache all files you already visited on your own computer. Maybe someone else already fetched the page you are demanding - this will speed things up.

Some internet providers don't like their customers to have full internet access or don't have enough free routable host addresses and therefore install a firewall. Most times on these firewalls a proxy is running, too.

1.4 What's new?

```
                Notes to Upgraders
==================

The most important notes to upgraders for the last few versions.
For the full revision notes take a look at the History section.
They are much shorter, however.

------------------------
0.13beta   ->   0.14beta
------------------------
```

- The new caching system is implemented and working. No more startup delays after crashes, no more long delay times with big caches, when a request does not return a valid document the old cache is kept, and more.

- The old cache format can be converted with the included updateCache.rexx script. Just use the installer script.

- New section
  Technical Stuff
  with some information about internals and the cache system structure.

- New httpresolve utility telling you which file belongs to which URL and vice versa. Needed for the conversion script as well.

- New options for HttpProxy (take a look at the
  options
  section:

```
  + [minnumreq NUMBER]     specify minimum number of available request
        slots for interaktive requests (and not for
        queued requests)
  + [keepbad]              keep document caches even when an error occured
        while talking to the server. Note that
        previously received (valid) caches are deleted
        with this option set.
  + [proxylocal]           do not cache requests sent to the local host.
        (usefull for local servers).
  + [noqueue]              do not start in queuing mode (a service URL
        exists to change this on the fly).
  + [nohttpproxyproxy]     do not use proxyproxy for HTTP requests.
        Only for FTP/GOPHER/WAIS this proxyproxy will be
        used.
  + [debug LEVEL]          specify amount of debugging output.
```

- default value for [delete SECONDS] is changed to about two years, as HttpProxy should not perform any deletes anymore. That is what HttpDelete is made for. The option will vanish sooner or later.

- new service URL
  mq*
  for changing the queuing mode.

- debugging system reviced. Now the amout of debugging output can be reduced by specifying a debuglevel. Type 'httpproxy ?' to get information about the debuglevels.

- HttpProxy sends old caches now instead of 'Your request is queued' for image files (dumb url extension file type recognition, though..). Note that this no more than an interim sollution to an old problem until the main structure of HttpProxy has been changed.

- Queued non-existant images will show a little graphic saying that they are queued. No more 'broken-image'-images...

- Queuing information is written on every queued request now. You will no longer loose your queuing information when you switch off your computer

```
  without stopping HttpProxy. The '@queue' file will get longer and longer
  that way, however. Note that the queuing system may not work properly
  with more than 64 entries right now.

- The installer script has been enhanced a lot.

- Some 'Goodies' have been added.


------------------------
0.12beta   ->   0.13beta
------------------------

- Please use the installer script to re-install HttpProxy or read the

                Installation
                 section carefully. Startup options have changed a lot and
  there are additional tools, too.

- Beginning with V0.13 hostname caching is introduced thus making HttpProxy
  fast in non-proxyproxy mode.

- Timeouts are implemented. A TIMEOUT
                option
                 exists for maximum delay time
  specification. Default is 10 mins.

- It is now possible to change between online and offline state while
  HttpProxy is running. This is controled by special URLs which are
  described in the
                Usage
                 section.

- The
                Usage
                 section is in fact a new section. The old Usage section is now
  called
                Startup
                , as this matches its purpose much better.

- The network protocol handler is now completely encapsulated and another
  network protocol handler for INet / AS225 / Surfer was added. HttpProxy
  determins on startup time the right protocol handler.

- The log file generation has been encapsulated, but some work has to be
  done here. The log file format will change another time. Be warned!

- No log file is generated any more when you don't specify a 'log'
                option
                .

- The default expire and delete times have been changed.

- A real support page has been installed on

  http://wwwcip.informatik.uni-erlangen.de/user/mshopf/httpproxy.html
```

1.5 HttpProxy - Features

```
                Features
========

First of all HttpProxy is a
                proxy
                 and does exactly what it is  supposed  to
do.

It caches all visited URLs and expires and deletes its cache files after  a
given  timeout  or  after an immediate second request (within 10 seconds by
default). This method will change in future versions.


HttpProxy has several basic configurations which can be changed on the fly:


1) proxy for http: URLs to remote machines

   In this mode only http: requests are accepted. HttpProxy does  not  have
   other protocols built in than HTTP.


2) proxy for all non-interactive requests (http:, ftp:, gopher:, wais:, ..)
   to another proxy (only caching)

   In this mode
                almost
                 all requests  are  possible  and  are  forwarded  to
   another proxy specified by you.

   This mode will be refered as 'proxyproxy' mode for obvious reasons.


3) proxy for already visited URLs in offline mode

   While you are offline (only sensible for dial-in connections, of course)
   you  are  able  to  browse  through  all  documents you already visited.
   HttpProxy will inform you whenever a page is outdated, but you  will  be
   able to receive this expired one by immedeately reloading the document.


4) queuer for new URLs in offline mode and auto spider in online mode

   While  you  are offline you are able to click on links or enter URLs you
   don't have visited already. Those URLs are queued and the next time  you
   are   online   these   pages   are  loaded  automatically,  up  to  four
   simultaneously.

   I plan to add an auto spider option that automatically gets all sub-URLs
   of a requested URL on request, but that is yet to come.
```

HttpProxy may have even more features in the future, but that depends very much on
you and your response
!

Please note also that this version is still
beta
software!

1.6 HttpProxy - Requirements

Requirements
============

- An Amiga® system

- On Amiga Kickstart 2.0 should be enough, but it is only tested on an A4000 with Kickstart 3.1 so far by myself.

- AmiTCP V2.2 or greater or INet / AS225 / Surfer V4 or greater installed and running.
  I was told that Miami works very well with HttpProxy.

  The AmiTCP V4.3 demo from the german 'Amiga Magazin' does *not* work! Get the AmiTCP V4.0 demo from Aminet instead.

- Some Mbytes of harddisk space (you don't want to run AmiTCP from floppy, do you?) for the cached data.

- Some Mbytes of memory. Memory usage dropped a lot begining with V0.11, AmiTCP, IBrowse and MUI will take away the rest, however...

- Ability to connect to a WWW-server ;-)

- that's all...

1.7 HttpProxy - Beta Version!

Beta Version!
=============

Note! This is still beta!

Though I will never claim any code of me (other than HelloWorld.c ;-) to be bug free, beta software is even less bug free...

However HttpProxy never ever crashed on my computer (well, 0.12 alpha 1 crashed once - but I think it was AMosaic's fault ;) and when it freezed it was always interruptable with Ctrl-C (Gee, the first >10Kbyte proggy I ever wrote that didn’t send any Guru at all to me B-). The latest version has been working fine on my computer and those of my alpha testers for over a week on release time.

You should always run it on a small partition, though, there’s a big chance it will invalidate your harddisk when any programm (maybe AMosaic?) crashes while HttpProxy saves its data to a cache file. But that problem can not be solved at all...

Be carefull on first time configuration. On startup HttpDelete deletes *recursively* all cache files and dirs that are older than specified. There must be special files present to make HttpDelete accept the directory as a cache directory, though, and the installer script is very intelligent in this part.

Please help me to find any bugs left in the code:

If you find HttpProxy crashing your computer or behaving strange on some URLs (please note the
Limitations
section) or in some environments, please
read the
Troubleshooting
and the
FAQ
sections, and when that fails, feel
free to
contact me
. EMail is prefered.

Please specify as much information as possible as this will greatly reduce debugging time. A statement of the form "HttpProxy doesn’t work" is really useless! And *please*, don’t forget to state the version you are using!

Note also that english is not my native language. I’m sure this guide file contains several severe misspellings and gramatical errors. Feel free to correct me (shame on me!).

Thank you for helping me to improve HttpProxy!

1.8 HttpProxy - Installation

Automatic Installation
======================

Please read the Installation Guide before using the installer script.

Unless you have some knowledge about TCP/IP and its startup files, you really *should* use the installer script. It doesn’t handle some very rare cases, though...

Note that the installer script is less intelligent in INet / AS225 / Surfer mode, because I don't have this package available for testing...

A special note: Never ever use the same cache directory for HttpProxy as for any browser supporting caching by itself. This will screw up both, HttpProxy and your browser!

Manual Installation

********** ATTENTION!!! *************
In order to update your old cache directory you will have to keep your old httpresolve executable. Do not delete it before installation.

The last beta version of HttpProxy did not have this utility, so you can safely ignore this warning for exactly this release.

That's the easy part:

Add the line

```
    http          80/tcp                ; World Wide Web
```

to your AmiTCP:db/services file.
In fact this is not necessary at all, but HttpProxy will send a warning message to your syslog when it can't find the http service.

Copy the according executables to any directory in your path (e.g. AmiTCP:bin)

There are several to choose from:

```
- httpproxy.000  Choose this one when you're running on a 68000 system
- httpproxy.020  Choose this one when you're running on a >=68020 system
- httpproxy.db   Choose this one when you want to have lots of debugging
     output (in order to send it to me =-)

- httpdelete.000 Choose this one when you're running on a 68000 system
- httpdelete.020 Choose this one when you're running on a >=68020 system

- httpfetch      (no choice available)
- httpresolve    (no choice available)
```

Remember not to delete your old httpresolve executable!

HttpProxy does not need a mathematical coprocessor because it performs no mathematical operations at all.

There are no different versions for different network stacks. HttpProxy determins by itself which protocol stack to use.

Note that debugging output slows the program down. Debugging files tend to grow very fast, too...

```
Now that’s the hard one...

Make  a  new  directory for cache files. You will specify this directory on
startup with the ’cache’
                option
                .
Make an empty  file  ’@dirurl’  inside  this  cache  directory  (e.g.  with
’echo >@dirurl’ and create two empty directories ’@temp’ and ’@trash’.
Copy all files from the ’msg’ directory into a new created ’@msg’ directory
inside the cache dir.

You should also save the current  cache  version  for  later  cache  update
scripts. For this invoke
   ’httpresolve >@cachever version’
inside the cache directory.


If you alread have a cache directory from older versions,  you  can  update
your cache with the supplied updateCache.rexx utility.

If you had an old httpresolve executable, copy it to ram:old_httpresolve.
Copy the new httpresolve executable to ram:httpresolve.

Set the current directory to the HttpProxy archive directory.

Now call the ARexx-Script by
   rx updateCache.rexx CACHEDIRECTORY
or by
   rx updateCache.rexx CACHEDIRECTORY dodeletes

The later one will delete all old cache files and directories. Use this one
when you are tight on space.


Now you have to tell your browsers how to contact the proxy.

-
                AMosaic
                -
                AWeb
                -
                Emacs
                 with WWW module
-
                IBrowse
                -
                Lynx
                -
                Mindwalker
                -
                Voyager
                -
                Weblink
                Then you have to start HttpProxy, whenever you start the network   ↩
                   protocol.
```

```
The
                options
                  depend  on  your  initial  startup  mode (online or offline),
however.

An Example: I use the line (all in one line, of course)

run AmiTCPUser:Cache/httpproxy >>Cache:ProxyCache/.HttpProxy.debug
    cache Cache:ProxyCache proxy proxy.rrze.uni-erlangen.de 80
    log Logs:HttpProxy.log offline get unread


Next you have to run HttpDelete every time you want to scan the directory
for expired cache files.

An Example: I run HttpDelete immedeately after starting HttpProxy by

run AmiTCPUser:Cache/httpdelete Cache:ProxyCache log Logs:HttpDelete.log
    delay 5 pri -1

You can do this by a cron job in regular intervals of course.


Next you should toggle HttpProxy's online state each time you  go  on-  and
offline by using the supplied
                special URLs
                .
You can do this with your browser, too, of course.

An Example: I use

AmiTCPUser:Cache/httpfetch service co1 quiet retries 5

*after* getting online and

AmiTCPUser:Cache/httpfetch service co0 quiet

*before* getting offline.
```

1.9 HttpProxy - AMosaic configuration

```
                Configuration for AMosaic
=========================

Just set the environment variable 'http_proxy' to

  http://localhost:PORT/

with PORT being the
                proxy port
                .

In the standard configuration (without 'port' option specified) this  would
result in
```

```
  http://localhost:8080/
```

When you run HttpProxy in proxyproxy mode you may set the environment variables 'ftp_proxy', 'gopher_proxy' and 'wais_proxy' to the same value.

1.10 HttpProxy - IBrowse configuration

```
                Configuration for IBrowse
=========================
```

Open the 'Preferences / Network...' window by using the menu.

Select the 'Proxy' cardfile.

Here you should specify

```
  http://localhost:PORT/
```

as http proxy with PORT being the proxy port.

In the standard configuration (without 'port' option specified) this would result in

```
  http://localhost:8080/
```

Then save the options.

When you run HttpProxy in proxyproxy mode you may set the ftp and gopher proxy entries to the same value.

1.11 HttpProxy - Lynx configuration

```
                Configuration for Lynx
======================
```

Add the following entry to the 'lynx.cfg' file:

```
  http_proxy:http://localhost:PORT/
```

with PORT being the proxy port.

In the standard configuration (without 'port' option specified) this would result in

```
  http_proxy:http://localhost:8080/
```

When you run HttpProxy in proxyproxy mode you may set the configuration entries 'ftp_proxy', 'gopher_proxy' and 'wais_proxy' to the same value, e.g.

```
  ftp_proxy:http://localhost:8080/
  gopher_proxy:http://localhost:8080/
  wais_proxy:http://localhost:8080/
```

1.12 HttpProxy - Weblink configuration

```
Configuration for Weblink
=========================
```

Sorry, I don't know enough right know about the coming Weblink AmiTCP network module (Sorry, Jesse, when I missed something ;-). Perhaps it is posssible to programm a new api module that interacts directly with HttpProxy. But that is yet to come.

1.13 HttpProxy - AWeb configuration

```
                Configuration for AWeb
======================
```

Open the 'Settings / Change Settings...' window by using the menu.

Select the 'Network 2: Proxy' page.

Here you should specify

```
  http://localhost:PORT/
```

as http proxy with PORT being the proxy port.

In the standard configuration (without 'port' option specified) this would result in

```
  http://localhost:8080/
```

Do not select 'limited proxy usage'.

Then save the options.

```
When you run HttpProxy in
                proxyproxy
                 mode you may set the  ftp  and  gopher
proxy entries to the same value.
```

1.14 HttpProxy - Mindwalker configuration

```
                Configuration for Mindwalker
============================

Sorry, I don't know enough about Mindwalker. Perhaps someone else  could  send
me a note?

Perhaps it is just the same as
                Voyager
                .
```

1.15 HttpProxy - Voyager configuration

```
                Configuration for Voyager
=========================

Open the 'Settings / Network...' window by using the menu.

Select the 'Proxies' cardfile.

Here you should specify

  'localhost'     as host and    'PORT'     as port

for http proxy with PORT being the
                proxy port
                .

In the standard configuration (without 'port' option specified) this  would
result in

  'localhost'     as host and    '8080'     as port.

Then save the options.

When you run HttpProxy in
                proxyproxy
                 mode you may set the  ftp  and  gopher
proxy entries to the same value.
```

1.16 HttpProxy - Emacs configuration

```
Configuration for Emacs
=======================

Sorry, I don’t know enough about Emacs and its Amiga port. Perhaps  someone
else could send me a note?
```

1.17 HttpProxy - Startup

```
                There are some tools provided for HttpProxy. Their usage  is   ↩
                   described  on
seperate pages:


                HttpDelete
                  Delete all cache files that are too old.

                HttpFetch
                  Yet another URL fetch  utility.

                HttpResolve
                  Resolve URLs and cache file names.


All known Options
=================

Usage and known debug levels (only debug version):
        httpproxy ?

httpproxy     [proxy PROXYHOST PROXYPORT] [port PORT] [cache DIR]
        [del SECONDS] [expire SECONDS] [reload SECONDS]
        [timeout SECONDS] [log FILE] [numreq NUMBER]
        [minnumreq NUMBER] [unread] [offline] [get] [keepbad]
        [proxylocal] [noqueue] [nohttpproxyproxy] [debug LEVEL]

(I know, this is no standard Amiga template...)

- proxy       PROXYHOST PROXYPORT
        Default: <nothing>
        Specifies a
                proxyproxy
                . If you don’t know  the  port  number,
        try both 80 and 8080. These are the standard values.
        It can be usefull to specify a  proxyproxy  even  in  offline
        mode as this will enable queueing of non-http requests.

- port        PORT
        Default: 8080
        Specifies the port HttpProxy should listen on. This value  is
        needed when you
                configure
                 your browser.
```

```
- cache       DIR
        Default: <current directory>
        Specifies the  cache  directory  you  created  on  installing
        HttpProxy.  The  program  changes its lokal directory to this
        place, so if you don’t run it  in  the  background  you  will
        notice some strange effects on Amigas. For instance the shell
        will think it is still in the old directory,  but  ’cd’  will
        prove that you are not.

- del         SECONDS
        Default: 2 years
        This will no longer be supported. Use the
                HttpDelete
                  utility
        instead. In fact, it does nothing in the current release.

- expire      SECONDS
        Default: 1 week
        Specifies the minimum age of cache files that are  marked  as
        invalid. Requests for invalid URLs are queued in offline mode
        and reloaded in online mode. The old invalid cache  file  can
        be viewed in offline mode by immedeately reloading the URL.

- reload      SECONDS
        Default: 10 seconds
        Specifies the maximum  (!)  number  of  seconds  between  two
        (identical) requests to be interpretated as a reload request.
        Used on an expired cache  entry  the  invalid  page  will  be
        presented  to  you. Used on a valid cache entry the page will
        be reloaded in online mode and queued in offline mode.

- timeout     SECONDS
        Default: 10 minutes
        Specifies a minimum timeout value. Timeouts are layed on both
        connections,  client  requests  to  HttpProxy  and  HttpProxy
        requests to remote servers. When either  is  not  transfering
        any  data  inbetween  the  specified  time, the connection is
        canceled.
        Due to implementation details the real timeout can be delayed
        up  to  one more minute than specified (or any other constant
        value specified on compile time).

- log         FILE
        Default: <no logfile>
        The logfile. Don’t specify if you don’t want one.

- numreq      NUMBER
        Default: 16
        Specifies the maximum number of request  slots.  The  default
        number  should  suffice  for  everything  but proxies used by
        multiple users (and perhaps usage of  good  ethernet  links).
        This  specifies  the  maximum  number of requests open at the
        same time.

- minnumreq   NUMBER
        Default: 12
        Specifies the minimum number of request  slots  reserved  for
        interactive requests. Only the remaining slots may be used by
        HttpProxy for getting queueud URLs. If you raise ’numreq’ you
        will  want  to  raise this number as well. Otherwise you will
        find yourself waiting for the queue fetching to finish.

- unread
        Specify this switch when you want interrupted transfers to be
        continued  by  the  proxy (e.g. you interrupt the transfer in
        the AMosaic window, but the proxy will keep getting data).

- offline
        Specify this switch  when  you  are  in  offline  mode  while
        starting HttpProxy. Can be changed while HttpProxy is running
        through a
                special URL
                .

- get
        Specify this switch when you want  the  proxy  to  auto  load
        queued  URLs.  Does  only work in online mode, of course, but
        may be spezified in offline mode, too. Can be  changed  while
        HttpProxy is running through a
                special URL
                .

- keepbad
        When this switch is  specified  cache  files  are  no  longer
        deleted,  when  an error occures while HttpProxy receives the
        URL. Changing the online state to offline is  treated  as  an
        error in this case, too.
        Note that you may loose a (valid) old cache  entry  when  you
        specify this switch. The old cache entry will be kept when an
        error occures during transmission otherwise.

- proxylocal
        When you are running a local HTTP server you can specify this
        switch.  In  that  case HttpProxy will not cache any requests
        sent to your local host. Note that only ’localhost’  and  its
        realname are recognized, no aliases.

- noqueue
        Starts HttpProxy in non-queuing  mode.  More  about  the  new
        queuing  mode  can  be  found  in  one  of  the
                service URLs
                        sections.

- nohttpproxyproxy
        When you do not want to use HttpProxy as a proxyproxy in  the
        general  case,  but  you  do  like to use FTP / GOPHER / WAIS
        requests, you can specify a proxyproxy *and* this option.  By
        that all HTTP requests will be sent directly to the specified
        host and all other (unknown) requests will  be  sent  to  the
        proxyproxy.

- debug      LEVEL
        With this option you can specify how much  debug  output  you
        want  to  have  included.  Some  options  produce pretty much
        output... All known debug options can be printed with
        'httpproxy.db ?'
        These values can be or'ed to enable them all.
```

1.18 HttpProxy - Usage

```
                Usage and special URLs
======================

When you are in offline mode and click on an  URL  that  is  not  currently
cached,  your request is queued and the specified URL is fetched as soon as
you get online again (only when you specify the
                'get' option
                 or reconfigure
the 'get' configuration - see below).

When you click on the reload button of your  browser  within  the
                'reload'
                time  limit,  HttpProxy  will  fetch  the  document  again, even  ↩
                    when it is
already cached. It must be exactly the same request in order  to  work,  so
you will have to press reload at least two times when the page contains any
inline graphics.


Online configuration is only possible from  localhost.  It  is  a  security
problem  and  it  doesn't make sense to reconfigure online state from other
hosts, too...
This will change as soon as document specific configuration is added.

For online configuration a special URL has been introduced. Its format is

  http://proxy.../xyz

with xyz beeing the configuration option.

You can use your browser to send configuration requests to  HttpProxy,  you
can  use  any  'httpget'-like  utility  or  the supplied
                HttpFetch
                 utility.
HttpFetch has a 'service' option which will construct the special  URL  for
you.

The configuration options consist of a type character, an option  character
and additional option arguments.

Right now the following type characters are supported:


                   c
                   Change the configuration.
                   m
                   Set the working mode.


An Example:

To switch in online mode the URL

  http://proxy.../co1

has to be sent to HttpProxy. Either use your browser or use  the  HttpFetch
utility:

  httpfetch service co1

You may want to specifiy the 'quiet' option, too.
```

1.19 HttpProxy - Special URLs type 'c'

```
Special URLs
============

Option Type 'c':   Change the configuration
===========================================

The following configuration options are supported:

co*          (* being one of '0' and '1')
        Switch online state. You can switch  into  online  mode  with
        'co1'  and  into  offline  mode  with  'co0'.  When  you told
        HttpProxy that it should get queued URLs when getting online,
        these  URLs  are requested as soon as you switch HttpProxy to
        online mode.

cg*          (* being one of '0' and '$^1$')
        Switch autoget state on or off. When you are online,  autoget
        mode  was  off  and  you  switch  it  on, all queued URLs are
        fetched automatically.
```

1.20 HttpProxy - Special URLs type 'm'

```
Special URLs
============

Option Type 'm':   Change the working mode
==========================================

The following mode options are supported:

mr*          (* being one of '0' and '1')
        Switch reload mode. With 'mr1' you can tell HttpProxy that it
        should  reload everything for which a request will be sent in
        the future. Use this e.g. for  image  cache  updates  or  for
        reloads in Voyager.
        Don't forget to switch this off after  it  did  what  it  was
        intended  to  do. If you use this mode while you are offline,
        all requests will be queued.

mq*           (* being one of '0' and '1')
        Switch queuing mode. When HttpProxy is not in  queuing  mode,
        it  will neither queue any expired entries nor any new links.
        Reload requests will still queue the current document, so you
        can safely check first whether you already have that document
        and afterwards decided to queue it.
```

1.21 HttpProxy - Tools

```
                There are some tools provided for HttpProxy. Their usage  is   ←
                   described  on
seperate pages:


                HttpDelete
                  Recursively scan a cache directory and delete all files  that
        are too old.


                HttpFetch
                  Yet another URL fetch  utility.  This  one  contains  special
        features  for  use with HttpProxy and supports the AmiTCP and
        the INet / AS225 / Surfer network stack.


                HttpResolve
                  Ever  wondered  which  file was associated with which URL and
        vice versa? With this utility you can  determine  that  in  a
        cache-version independend way.
```

1.22 HttpProxy - HttpDelete tool

```
                HttpDelete usage
================

HttpDelete scans recursively a cache directory for invalid cache files  and
deletes all expired files and directories. Use with care!

In order not to delete any wrong directories, a file '.iscachedir'  has  to
be there. Otherwise HttpDelete won't start.
Right now only Files starting with '_' and '@' are deleted anyway, but this
will change as soon as an other cache system has been implemented.

HttpDelete may be interupted savely with CTRL-C.
```

```
Usage:    'httpdelete ?'
Defaults: 'httpdelete'
Template: CACHEDIR=DIR/A,TIME/N,LOGFILE=LOG/K,EXBUFFERSIZE=EXBUF/K/N,
    CHECKONLY=CHECK/S,DELAY/K/N,PRIORITY=PRI/K/N

- cachedir    DIRECTORY
        Specifies the cache directory to be scaned.

- time        SECONDS
        Specifies the  minimum  age  in  seconds  before  a  file  is
        removed.  Set this to the same amount as you specified to the

                'del'
                 option of HttpProxy.

- logfile     FILENAME
        Should be obvious. For the format of the logfile take a  look
        at the source.
        Use 'console:' when you just want to view the output.

- exbuffersize BYTES
        The size of  the  ExAllBuffer  per  directory  level.  Larger
        sizes  than  the default don't speed up HttpDelete that much,
        but influence  HttpDelete's  Ctrl-C  reaction  time.  Smaller
        amounts  make  HttpDelete  a  bit  slower  but  reduce memory
        consumption.

- checkonly
        Don't perform any deletes, just write log entries. Without  a
        specified logfile option this doesn't make much sense...

- delay       TICKS
        1/50 second intervalls to be waited after each ExAll()  call.
        Mostly usefull with not too high values for EXBUFFERSIZE (not
        more than the default at least) in  server  environments,  to
        run HttpDelete in the background. Use low priority, too.

- priority    PRIORITY
        Change  the  runtime  priority.  No  change  to  the  current
        priority is done, when this option is not specified.
```

1.23 HttpProxy - HttpFetch tool

```
                HttpFetch usage
===============

HttpFetch is yet another URL  fetch  utility.  This  one  contains  special
features  for  use  with  HttpProxy  and supports the AmiTCP and the INet /
AS225 / Surfer network stack.

HttpFetch can *not* be interupted with CTRL-C right now!
```

```
Short description: httpfetch
Usage:             httpfetch ?
Template:          URL/A,FROM=PROXY/K,TO=FILE,RETRIES=RETRY/N/K,
       HEADER=FULL/S,SERVICE/S,DISCARD=QUIET/S

- url         URL
        Specifies the URL to  be  retrieved  or  the  service  to  be
        performed. The URL is fetched from the host it resides unless
        it is a service or the 'from' option is specified.

- from        HOST   or   HOST:PORT
        Specifies a proxy host  address  to  which  the  url  request
        should  be sent. You may specify another port number than the
        default.

- to          FILENAME
        Save the URL contents into a file. Unless you use this option
        the URL contents are printed on stdout.

- retries     NUMBER
        Performs a retry for the connection  after  a  delay  of  one
        second after each try.

- header
        Save / print the  header  information,  too.  HttpFetch  only
        sends  HTTP/1.0  requests  (I  don't know whether all servers
        support HTTP/0.9 requests any more...) and skips  the  header
        part in standard mode.

- service
        The specified URL is in fact a service request.  The  URL  is
        constructed  by  HttpFetch itself. This is only for HttpProxy
        support and explained in the
                Usage
                 section.

- discard
        Discard the output. This is faster than redirecting output or
        saving to nil:


Some example invokations:


- httpfetch service co0 quiet

  Sets HttpProxy on the local host in offline mode by requesting the URL
  'http://proxy.../co0'. Output is discarded as you are not interested in
  the confirmation message of HttpProxy.

- httpfetch http://www.wuarchive.com/

  Gets the URL directly from www.wuarchive.com and prints the output on
  stdout.

- httpfetch http://www.wuarchive.com/ to t:wuarchive_index header retry 5
```

```
  Gets just the same URL but outputs it into the file t:wuarchive_index,
  saves the header, too (just try to get the difference), and tries five
  times to get the URL, in case any error occures while transfering.

- httpfetch http://www.wuarchive.com/ from localhost quiet

  Instructs a local running HttpProxy to get the URL and save it for
  further offline reading. Note that the document may be cached already. It
  is not fetched from the net then!

- httpfetch http://www.wuarchive.com/ from localhost:8081 quiet

  The same, only HttpProxy is running on port 8081 and not on the default
  port 8080.
```

1.24 HttpResolve - URL to filename resolve tool

```
HttpResolve usage
=================

Ever wondered which file was associated with which URL and vice versa? With
this utility you can determine that in a cache-version independend way.

Usage:             httpresolve ?        (inside cache-directory only)
Template:          URL/K,CACHE=FILE/K,TO/K,VERSION=VER/S,SAVE=WRITE/S

- url         URL
        Convert URL name to filename. The corresponding filename  for
        the  given  URL  is  printed  to stdout. Note that additional
        directories may be created. These directories will  be  empty
        and deleted with the next HttpDelete run, however.

- cache       FILENAME
        Convert the given filename to its corresponding URL. It  will
        be  printed  to  stdout. Note that all cache directories upto
        the given filename and additional needed hash files  have  to
        exist for HttpResolve to run successfully.

- to          FILENAME
        Print the name into the given filename, not to stdout.

- version
        Print the implemented cache version. This can be compared to
        the magic '@cachever' file inside the cache directory.

- save
        Copy stdin into a new  created  cachefile.  This  only  makes
        sense when specified together with the url option, of course.


Some example invokations:
```

```
- httpresolve url http://wwwcip.informatik.uni-erlangen.de:80/user/mshopf/ ↩
    httpproxy.html

  Will output the relative file name
     http/@!c2d57a94/user/mshopf/httpproxy.html@

- httpresolve cache http/ftp.uni-erlangen.de!J80/~ftpamiga/@

  Will output the corresponding URL
     http://ftp.uni-erlangen.de:80/~ftpamiga/

- ram:httpresolve <Cache:proxycache__old/_31f65717.000001ba SAVE URL "http://ftp. ↩
    rrze.uni-erlangen.de:8080/pub/amiga/aminet/info/www/recent.html"

  Will create a new cache file with the contents of the old cache file.
  BTW - this is an  example  invokation  created  by  the  updateCache.rexx
  script.

- httpresolve version

  Will just output
     2.4
  right now.
```

1.25 HttpProxy - Troubleshooting

```
                Troubleshooting
===============

This section is built upon questions I received during the  last  releases.
It  is  no  way  complete  but  I  will  add more answers as I receive more
questions :)

Most of the following questions should be obsolete now because of the new
intelligent installer script. :)

---------------

Q: Is there a HttpProxy home page in the WWW?
A: Yes. It can be found on
   http://wwwcip.informatik.uni-erlangen.de/user/mshopf/httpproxy.html

Q: HttpProxy suddenly terminated with a output line
   'Assertion (bla) failed in file 'foo.c' on line xyz'. Is that a
   restriction of an unregistered version?
A: That's a bug :] When the line is not mentioned in my
                history page
                ,
   please drop a mail to me containing the assertion and the line number.

Q: HttpProxy works fine when I'm online but I simply can't access it when
   I'm offline...
A: It is not sufficient to start AmiTCP:AmiTCP for the offline mode. There
   are three solutions:
   1) Configure another startnet (call it startoff or the like) for offline
```

```
      mode.
   2) When you go online *after* starting AmiTCP:bin/startnet (e.g. by a
      slip-dialer) just work without starting the dialer.
   3) Go offline immedeately after startup ;) by typing
      'offline *sana2*.device 0' with *sana2* being your favourite network
      device (e.g. rhcslip or ppp).

Q: HttpProxy works fine when I'm online but I always get the error
   'Proxyproxy host unreachable' or 'Host unreachable' from the proxy when
   I'm offline.
A: You have to start HttpProxy with the 'offline' option. When you started
   the TCP/IP-Stack in online mode and went offline afterwards you should
   send the URL 'http://proxy.../co0' to the proxy (with either your
   browser or by executing 'httpfetch service co0').

Q: After I went offline I sent a Ctrl-C to HttpProxy (it terminated
   correctly) and I wanted to start it again in offline mode, it sometimes/
   always refuses to start and says that the port is already occupied.
A: I never found out why, but TCP/IP requires the port to be occupied for
   the next few seconds after closing a server port (about 20-30 seconds).
   That's nothing special, at least on Sun workstations the same happens to
   me. Just wait a while and start it again...
   But you'd better use the new
                service urls
                 for changing the online state
   on the fly. See above.

Q: The other way round: HttpProxy works fine when I'm offline (I can queue
   requests and so on), but I can't get any pages when I'm online. The
   pages I queued won't be received, too.
A: 1) You don't have a routable IP-address. In that case you used to get
      http-pages not directly from the net but from your providers proxy.
      In this case you will have to specify your providers proxy with the
      'proxy' option, e.g. if your provider's proxy was
      'http://proxy.my.provider.de/' you would specify the option
      'proxy proxy.my.provider 80' (80 is the standard http port).
   2) In AmiTCP:db/services you added the line
      http 8080/tcp
      which is wrong. When it is present it should be
      http 80/tcp
   3) You went online *after* you started your TCP/IP-stack in offline
      mode. The you will have to send the URL 'http://proxy.../co1' to
      HttpProxy in order to tell him that you are now online. Use your
      browser for that or execute 'httpfetch service co1'.

Q: I just can't get HttpProxy work. I configured my browser
   (e.g. $http_proxy), I started HttpProxy.db with the correct port
   (e.g. 8080) and I get an 'Server is down' or something similar upon
   every request. No debug output at all happens when I try to get an URL.
A: 1) Try
      'askhost localhost'
      When that fails, an entry '127.0.0.1 localhost' is need in your
      AmiTCP:db/hosts file or the loopback-device 'lo0' wasn't configured
      (check your AmiTCP:bin/startnet).
   2) When it succeeded try
      'ping localhost'
      When you don't get continuing replies, something in your AmiTCP setup
      is strange (sorry, can't be more specific... call a guru...).
      Perhaps you only started AmiTCP:AmiTCP instead of AmiTCP:bin/startnet
   3) Try 'letnet localhost 13' and check whether the current date is
      returned.
   3) When that succeeded try
      'letnet localhost 8080' ('letnet localhost PORT' respectively)
      after starting the debug version HttpProxy.db (without output
      redirection!). When you get a 'Connection refused' and no debug
      output at all (after some initial stuff), you configured AmiTCP on
      installation the wrong way:
      HttpProxy is an active server. For this it needs AmiTCP to be
      configured to accept local servers. On installation of AmiTCP V4.2
      you will be asked, whether you intend to use server programms.
      You will have to answer with *yes* (AmiTCP does not configure the
      loopback-device otherwise...).
      Note that AmiTCP V4.3 demo from the german 'Amiga Magazin' is not
      able to support local servers. Get the AmiTCP V4.0 demo from Aminet
      instead.
      When you didn't install AmiTCP V4.2, the only advise I can give is to
      check your AmiTCP:db/inet.access manually (perhaps you disabled
      yourself from accessing the proxy port?).
      When you receive no output from the 'letnet', try pressing return.
      When you get a short message telling you that this is no valid request,
      your browser setup is not correct, it doesn't know about HttpProxy
      being there, but everything else is working.

----------------

When everything else fails...
                mail me
                ... :(
```

1.26 HttpProxy - FAQ

```
                Frequently Asked Questions
==========================

This section is built upon questions I received during the  last  releases.
It  is  no  way  complete  but  I  will  add more answers as I receive more
questions :)

---------------

Q: Is there a HttpProxy home page in the WWW?
A: Yes. It can be found on
   http://wwwcip.informatik.uni-erlangen.de/user/mshopf/httpproxy.html

Q: What is the newest available version of HttpProxy?
A: Check the support page. The most recent version can be downloaded
   from there, too.


Q: I cannot get HttpProxy work together with IBrowse. In offline on most
   links IBrowse just returns an error message.
```

```
A: In fact HttpProxy works fine, but IBrowse refused to show the queuing or
   expire notify messages before version 0.133 (17.07.96). This one or any
   later version can be downloaded from the IBrowse home page.

Q: I can’t get any pages that need authorization.
A: This bug is already known, there exists a workaround, however.
   Just press ’Reload’ immedeately after you got the message that
   ’authorization has failed’ / ’you are not allowed to view this page’.


Q: By pressing the ’Reload’ button I always get only the already cached
   page, HttpProxy does not reload the page from the net as intended.
A: You have to press the ’Reload’ button no more than about 10 seconds
   after the last request (that’s at least the default value). When the
   page is too big, this may be impossible. In this case interrupt the
   transfer early and press ’Reload’ again.

Q: It still does not work on most pages.
A: Try pressing ’Reload’ two times in succession. Most pages contain
   graphics which result in additional requests. HttpProxy will only reload
   a page when it receives exactly the same request two times in a row.

Q: It still does not work... BTW - I’m using Voyager.
A: Voyager reloads all graphics from a page by pressing reload. Even with
   disabled automatic graphics loading it requests the background...
   The only possible sollution is to temporary disable the cache. For this
   set HttpProxy into reload mode by sending the
                service
                 ’mr1’ via the
   special URL ’http://proxy.../mr1’ or with the
                HttpFetch
                 utility. Don’t
   forget to put HttpProxy back into normal mode by the service ’mr0’
   afterwards.


Q: HttpProxy is very slow. I use Executive, by the way.
A: You should either raise HttpProxy’s priority or exclude it from the
   Executive sceduling list. It needs very little CPU time, but when it is
   not resceduled when data is about to be received, it will block
   everything else.

Q: When I press on an image link in online mode I get the page, but when I
   do that in offline mode I won’t get the page I already visited.
A: You might be following a map link. These links cannot be cached as their
   contents may vary for every mouse position while clicking (you won’t get
   exactly the same position while you were online...).
   That’s one reason why I hate maps... :-(
   The only possible sollution is to add a bookmark entry after receiving
   the page and to only use the bookmark entry in the future. By that you
   get exactly the same mouse coordinates every time, of course.

----------------

When you stll have questions...
                mail me
                ... :)
```

1.27 HttpProxy - Things left to do

```
Todo - List
===========

in (some) order of importance for me :


- internal main program reorganization

- better logfile output

- usage of asynchronous filesystem calls (async.lib)

- auto contact interrupt when it is clear that the cache is valid
  (HttpProxy will contact the specified host on request and use the cache
  when the Modified: field shows, that the cache is valid. Otherwise it
  will get the document again.)

- 'If-modified-since:' requests.

- check of Expire:, Pragma: fields

- Cachebrowser for simple delete, queue and validate operations on existing
  cache files.


- athentication cache.

- configuration files (argument list gets longer and longer)

- user configurable document specific expire times (e.g. don't expire any
  URLs that match #?.jpg or #?.gif)
  That includes URL patterns that will never be cached (e.g. #?.lha)

- better handling of documents that are expired (more options for the user)

- special URL requests to inform user about queued and auto-requested
  documents (and perhaps structures) and cache information

- user configurable document specific online state (e.g. local server) and
  proxyproxy mode

- autoget of user defined URLs when they are expired (only on startup)


- access control (at least a simple one)

- automatic or semiautomatic detection of online state and adapting
  behavour accordingly

- startup of script commands after all queued files were received
- host name and URL aliases.


- save full URL requests in url cache file, send full URL auto request
  (may be this will never implemented as it can be a security problem)

- only keep on caching interrupted requests, when > xx% (on Content-Length:
  known) or > yyy Bytes are already read

- Priority routing for prefering interactive requests (when possible)


- auto spider

- automatic addition of Content-Length: fields wherever possible

- caching POST method

- other protocols (ftp, may be gopher)

- Specification of expire and delete timeouts in seconds / minutes / etc... /
  years.


- reload on immedeate second request needs work for cases where multiple
  recipients are involved

- change of inline links to file://... links when they are already in the
  cache
```

1.28 HttpProxy - Bugs and Limitations

```
                Known Bugs and Limitations
==========================

- The POST method is not cached right now. Luckily most forms use the GET
  method and normal documents are always fetched with the GET method.
  If your browser tells you that you sent an invalid request or the host
  couldn't be contacted, take a look at the source of the current document.
  You may find a 'action=post' tag inside a form description...

- The caching file system should be local (not nfs  mounted),  because  the
  cache file descriptors are not included in the select() list  (the  amiga
  version of select() doesn't support standard file descriptors here =-( ),
  and  because  of  this  the  file  operations  are  supposed  to   return
  immedeately or at least very fast. Asynchronous IO is planned to be
  added soon.

- Don't specify any URLs that would contact the proxy itself. It will lock
  up, because the URLs are not checked for this and:

- Error messages are cached as standard cache entries right now (they
  should be valid only once in offline mode and totally invalid in online
  mode)
```

```
- Requests that need Authorization are not processed correctly right now
  because of the cached error messages. Just enter your user name and
  password and press Reload immedeately after getting the error message.

- Some untested cases...

- Queued requests will only contain the URL, user-agent: and accept: */*
  (see
                Todo list
                )
```

1.29 HttpProxy - Compiling

```
Compiling
=========

Full source is provided for recompiling.

HttpProxy and its utilities were compiled with SAS C V6.56 with  near  data
and  code,  full  optimizing and no stack check and/or extension code (they
should never ever need more than 8 Kbytes of stack) and the AmiTCP V2.2 and
the current INet Api includes. A makefile is supported with several special
targets. No network link libraries are needed.

You won't need the rudimentary FIXTIME compiler define option any  more  as
the  network  Api library calls are outside the main source now. It will be
removed soon.

HttpProxy moves more and more to an Amiga-specific  solution  as  I  didn't
receive  any  requests  for  Unix-compatibility.  Well, these machines have
enough proxies to choose from...

The original source code is not the best I've ever written however (in fact
it was a four days hack plus additional bug fixes and enhancements...). But
it moves to a real program more and more :-)

If anyone wants to contribute some source to this project, please mail  me.
I  am open for any additional sources. They may take some time to be merged
into HttpProxy, however. Please note that the main source will  be  changed
*a  lot*  for  the  next  version. The network and cache modules are almost
complete right now.

One thing that I am really looking for is a network module for  Envoy.  Any
envoy programmer out there willing to do this? :-)
```

1.30 HttpProxy - Technical Stuff

```
                Technical Stuff
===============

This section is quite new, so maybe you do not yet find everything you want
```

to know about HttpProxy's internals here. In case you are suspicious about something just ask me.

If you want to know something about HttpProxy's internals, here you can find some information about what the
source modules
are used for.

Maybe you are just curious what all those fancy files in the cache directory mean? There is a big chapter about the
cache directory structure
and I hope you find some answers there.

1.31 HttpProxy - Modules

Source Modules
==============

This section is yet to be written...

1.32 HttpProxy - Cache Directory Structure

Cache Directory Structure
=========================

All cached URLs are saved in the cache directory in a hierarchical structure, most times the path reflects the URL in a 1:1 manner. Thus

```
  http://host.my.domain/bla/foo.html
```

should be saved in the cache directory Cache:ProxyCache as

```
  Cache:ProxyCache/http://host.my.domain/bla/foo.html
```

However, this is not possible with standard AmigaOS file system handlers. Additionally, no files needed for HttpProxy itself (e.g. queuing information) could be saved inside the cache directory and you could not distinquish between files and directories (e.g. /bla/foo/ <-> /bla/foo). And AmigaOS files are case insensitive with a maximum of 32 characters per partial path while URLs are case sensitive and have no size limit. Thus some characters are escaped and additional rules are added. This results in the above URL being saved in

```
  Cache:ProxyCache/http/host.my.domain/bla/foo.html@
```

The
rules for generating a file or directory name
are available, too. For
the following files it is only neccessary to know that inside the cache directory no directories or files beginning with '@' will be generated by

```
these rules except hash files beginning with '@@' and '@!'.

HttpDelete will only scan and delete directories  and  files  that  can  be
generated  by  these  rules.  Thus  the  following files will never ever be
touched. The only exception is the file '@dirurl' which will be deleted  by
HttpDelete every time it deletes the according directory.

Special files and directories
=============================

- @cachever   This file contains the current cache system version. It is
        used by the updateCache.rexx script to check the HttpResolve
        version and to check whether an update is neccessary or not.
        HttpProxy does not notice this file at all.

- @dirurl     This file contains the partial URL name connected with each
        directory. It is used for fast reversal URL resolving
        (HttpResolve). Though its content is redundant, it is needed
        for the resolving process right now.

- @queue      This file is written by queue.c on shutdown and on every
        queuing operation. It contains information about all queued
        and dequeued URLs. More information about the queue file
        may be obtained from the first few lines of the source.

- @msg/       This directory contains various files that will be sent by
        HttpProxy on certain events. They have to contain the full
        HTTP header.
        The only used file inside this directory upto now is
        'queued.gfx' which is sent for queued nonavailable graphics.

- @temp/      This directory is used for temporary files, e.g. cache files
        while they are received (when there is an old valid cache
        file existing in the main cache area). After a successfull
        load the existing old file is replaced with the temporary
        file.

- @trash/     Sometimes a old existing file cannot be removed because it is
        kept open for another connection. In that case it is moved
        into this directory. HttpDelete will scan that directory and
        delete all files herein. HttpProxy will also scan this
        directory on startup.
```

1.33 HttpProxy - Naming Rules

```
The Rules for generating a file or directory name
=================================================

1 The URL is divided into partial file or directory  names  and  each  part
  is  treated  seperately. When a partial name is larger than 30 characters
  or becomes larger than 30 characters in step (2), steps (4) is performed.
  When the path is already too long (MAX_PATH_LEN) or divided into too many
  partial names (MAX_FILE_DEPTH), step (4b) is performed.
```

The protocoll name and the host name parts are treated specially, because the former removes the ’://’ part of the URL and the later is treated without amibiguity checks (step (3)) (host names are case insensitive).

2 All invalid characters are escaped by ’!’ and an additional valid character. The possibility of an added ’!’ as the last character of a file or directory is reserved for future use.

Invalid characters are by definition:

’!’ and ’@’	(special meanings)
0-31, 127, 128-159, 255	(control characters)
32, 34 (’"’), 160	(space, double quote, shift space)
’:’, ’/’	(the later one is *never* used directly)

The following characters are *not* escaped by now. Perhaps they should...
’;’, ’*’, ’?’, ’`’, ’#’, ’%’
Thus using standard dos commands on cache files may be difficult. Try putting the cache filename into double quotes.

Finally, a ’@’ character is added to the partial name for the last partial name (the final file name) only.

3 The resulting name is checked for amibiguity: When the file/dir already exists, it is checked whether they are completely equivalent. With this check case sensitiveness and ISO-8859-1 conformance is guaranteed.
When the check fails, steps (4) and (5) are performed, otherwise they are skipped

4 A hash file entry is computed. It is computed in HashPartial() in cache.c by a standard method also used for compilers. Right now the hash value is computed over the full URL, not only over the partial name, but that may change.

Finally, a ’@’ character is added to the partial name for the last partial name (the final file name) only.

Two files will be created as soon as the cache file is opened.
One named ’@@xxxxxxxx’ with xxxxxxxx being the hexadecimal printed hash value, containing the complete URL upto this partial name.
The other named ’@!xxxxxxxx’ containing the document / the directory itself.

4b When the path is already too long or divided into too many parial names, a hash value is computed from the full remaining URL. A ’@’ character is added to the file name.

5 (Another) ambiguity check is done: The hash file ’@@xxxxxxxx’ is opened and its contents are checked with the current URL. Only when this check returns successfully, the hash file is used, otherwise the URL cannot be cached and will only be proxied. But that is very unlikely to happen.

6 The next partial name is processed when there is any one left
-> step (2) or (4).

1.34 HttpProxy - History

```
History
=======

Here you can find the most current text version of

http://wwwcip.informatik.uni-erlangen.de/user/mshopf/proxy_history.html

and the revision log of the main source file.

All sources have their revision log included in the header comment.

Anyway, please check the history page in in its original place (use any web
browser  for  this)  before you send a bug report to me. Perhaps the bug is
already known to me :)
```

1.35 HttpProxy - Disclaimer

```
Disclaimer
==========

In short:

I was not, I am not and I will be not be responsible for anything  this  or
other programs written by me or by others do or don’t do.

It’s your onw risk, to use, not to use, to  copy,  modify,  keep,  archive,
delete  this  program or other data or to do anything that can be done to a
program or to anything else with this program or with anything else.

Ufff...

And note it’s still beta!

However it works fine for me =-)


THERE IS NO WARRANTY FOR THIS PROGRAM TO THE EXTENT PERMITTED BY APPLICABLE
LAW.  EXCEPT  WHERE OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDER AND/OR
OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT  WARRANTY  OF  ANY  KIND,
EITHER  EXPRESSED  OR  IMPLIED,  INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR  A  PARTICULAR  PURPOSE.  THE
ENTIRE  RISK  AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.
SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST  OF  ALL  NECESSARY
SERVICING, REPAIR OR CORRECTION.

IN  NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL
ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY REDISTRIBUTE  THE  PROGRAM
AS  PERMITTED  ABOVE,  BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL,
SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING  OUT  OF  THE  USE  OR
INABILITY  TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR
DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD  PARTIES
OR  A  FAILURE  OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF
```

SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

1.36 HttpProxy - Copyright and Distribution

```
                Copyright and Distribution
==========================
```

HttpProxy is Copyright ©1995 by Matthias Hopf (Yes, that’s me B^)

All Rights Reserved.

This program is EMailware , *almost* freeware.

It is *not* public domain. The source is, though provided, still copyright by me.

You may modify or reverse engineer (though not necessary ;-) the program for your own use or for others as long as you don’t charge anything for it. But the original copyright messages in the code must not be changed! Don’t redistribute changed versions of the program without my explicit permission (a pgp signed emailed permission will be sufficient for this program).

The archive may only be distributed in unmodified form. No files may be added, changed or removed. That includes .displayme and similar files from mailboxes. Noone may charge for this archive more than normal media and duplication fees. Distribution is allowed in all forms except where noted.

Urban Mueller is explicitely allowed to include this program in archive or in ready to run form on any Aminet CD. Other parties have to contact me prior to including the program on CDs in any form or on disks or ftp sites in non-archive form.

Authors of Webbrowsers are welcome to contact me in order to get a redistribution license for HttpProxy along with their product for a minimal fee (in fact I just want a full version of your browser :-) .

All legal matters are subject to change without notice.

Written permissions by me can override all Copyright, Distribution, Disclaimer and Licence notes stated here.

For this program pgp signed EMails have the same status as written notes.

1.37 HttpProxy - License

```
                License
=======

You are hereby granted to use HttpProxy when you agree and  only  when  you
agree on the following conditions before and while using it:

- You will use the program at
                your own risk
                .

- You accept the
                Copyright and Distribution
                 restrictions.

- You will send an EMail notice to
                me
                 that  you  are  using  my  program.
  Snailmail and postcards are welcome even more, of course, but bug reports
  are prefered by EMail.

  Come   on,  a  line  'I'm  using  your marvelous/great/nice/buggy/sh*ting
  HttpProxy' would suffice though more would be nice. :)

  As you have access to a WWW-Server it normally should be  no  problem  to
  you to send a short EMail to me...

- You  will send an EMail notice to me specifying the approximate number of
  hosts and users when you install HttpProxy on a system on  which  several
  users have perhaps simultanously access to the proxy or several hosts are
  involved. Note that multiple users may result in some problems right now.
  However, some providers already use this proxy right now. =^)

Don't  be  afraid  to tell me that you don't like or can't use HttpProxy. I
just want to hear your *real* opinion. But I sincerely hope you like it.
```

1.38 HttpProxy - Acknowledgments

```
Kind regards and thanks are flying to:
======================================

- Michael Hettenbach
- Peter Wlcek
- Rene Hexel
- Carsten Becker

  for Alpha-Testing

- Martin Goebel
- Bruno Barbera

  for providing additional ARexx-Scripts (take a look at the Goodies
  directory :)
```

- Osma Ahvenlampi

  for providing me their INet addaptions of HttpProxy thus speeding up the
  porting process

- Fionn Behrens

  for providing me his temporary fix for compiling HttpProxy with the
  broken AmiTCP4.0 time() / stat() Api. It is not necessary any more,
  though.

- Rene Hexel
- Frank Copeland

  for additional guide information (Alynx config).

- Osma Ahvenlampi

  for his suggestion to use async.lib for cache i/o (never heard of it
  before).

and all guys and girls in the world supporting EMailware =-D
(I really got more EMails than I ever thought I could read)

Long live Amiga (and not Commodore)!

[I said that in 1988 :-]

1.39 HttpProxy - Support

```
                Support
=======
```

Remember: HttpProxy is EMAILWARE!
So go right now into your favourite mailer and send an EMail to
me
.

I’m open for bug reports, likes, dislikes, opinions, improvements.

HttpProxy now has a support page:

  http://wwwcip.informatik.uni-erlangen.de/user/mshopf/httpproxy.html

Be sure to check the history page (accessable from the support page)
immedeately before sending bug reports or feature requests to me. May be
the noted bug or requested feature is already known to me :-)

This program will only develop the way *you* like when *you* send your
comments about it to me.

And *please* - always state the HttpProxy version you’re using...

```
There are several
                things to do
                , but order is not fixed. I just sorted  all
features that came into my mind like I think they are important.
```

1.40 HttpProxy - Author

```
Author
======

Matthias Hopf
Neckstraße 4
91052 Erlangen

email: mshopf@informatik.uni-erlangen.de

I’m a student of computer science in Erlangen, Germany.

Feel free to EMail or to snailmail me =-D

You’re welcome on my WWW home page.


Please contact the history page from

  http://wwwcip.informatik.uni-erlangen.de/user/mshopf/httpproxy.html

immedeately before sending  bug  reports  or  feature  requests  concerning
HttpProxy to me. May be the noted bug or requested feature is already known
to me :-)

And *please* - always state the HttpProxy version you’re using...


-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6ui (Amiga)
```

```
mQCNAi7TlyUAAAEEAOyEU5wh5wNu1Pu5w+UGDPidBMlg3pPWXjkZf9vWGLu4vX14
DgAMwiGBcJlWE4KkiRabY9WmZdBjhuy2ECftZn8UBAxxxItgkN3pe8HObI7KWCdn
/ctlX9bzUymnAAZ46trFk6A3mwb6j5tYjKQJ14WZUPwDB5zsWD9QXspNMJEBAAUR
tDVNYXR0aGlhcyBIb3BmIDxtc2hvcGZAY2lwLmluZm9ybWF0aWsudW5pLWVybGFu
Z2VuLmRlPokAlQMFEDGSJ4C44C5kpvwluQEBPtcD/3A3ULRi50vng2SYDS1mZH05
S3pewyvIE75M+AK5Rb31P1xipYd27/clVHWzx9pgRrPt7xUJ46dqEKNcMv8O1kpr
FOg55PGrcg5gVkXmtFsbYa7PhkllV/AGXoyUbuXgWFK3fodBqi7iHoe1TJB9/Sw0
dXJ/7ELF/kubJpQkwGphiQBVAwUQMQzmq/Avc1PolDZ1AQExtwIAhNzR/6i6n7tU
/5OkbEqXHCQfhKsX3IiJWApEfo22X+j5a7vLxav0A/yGJv+hfm0YzGIE8+vpLXSm
2h0PZQRn6okAlQIFEDEF58o/UF7KTTCRAQEBissD/0uFgv5vLcLFA8KV0wHIPO2q
s5Ssa0UZFtBN9TRGYM5BGI2bs8Em6gZQFmPh/M0QiMvq3X4C2fpgPIS+KF3a504w
PM6oR4DNwa987aVnKn8KfGxD3cTDH7eg9RhaqAiY3iPPZp33i3AVNtPmmolWnhZr
ACz6/7gOtDGezUgDPyOEiQCVAgUQLz5Cb/kX5t+lybPVAQERmgQAnSAh5XhcAy9E
K6/qbynftZNZ+DS6rN5FdVlpZ/gkt6nfDOuVeWO2fdCFNdpusLAYuD1BcQrI+xkI
aUiave7Bydz3QuML0vaeZFVjMIUCC27u2Rb/EA8tTaIiLJy7pedRaYHkXjM0NMFs
iYA2GiIZWGayuGg/JdMtlw7JkhroSVE=
=/10d
```

```
-----END PGP PUBLIC KEY BLOCK-----


--
_   //             |   Matthias Hopf - "Hoeppel"   |    _      __
\\ //    Amiga     |  student of computer science  |  _|cience |-iction
 \X/ by conviction |      in Erlangen/Germany      | by belief in Future

 EMail:  mshopf@informatik.uni-erlangen.de
 Aminet: ftpamiga@epix.rrze.uni-erlangen.de
 WWW:    http://wwwcip.informatik.uni-erlangen.de/user/mshopf/
```