# fromfilter

# ${\bf Kostas\ Koukopoulos}$

# August 18, 2002

#### Abstract

from filter is a electronic mail filter using the libmilter API from sendmail. Its purpose is to prevent misrepresentation and impersonation from happening inside an organisation.

# Contents

1	Intr	roduction	2			
2	Copying					
3	libmilter API code					
	3.1	callback prototypes	3			
	3.2	envelope	4			
	3.3	header	6			
	3.4		9			
	3.5	filter description (struct smfiDesc)	12			
	3.6	struct PrivData declaration	13			
	3.7	cleanup	15			
	3.8	includes	16			
4	LDAP code					
	4.1	outline	16			
	4.2	ldap includes				
	4.3	types				
	4.4	globals				
	4.5	init_ldap	18			
	4.6	query_uid	20			
	4.7	attribute handlers	23			
	4.8	foo				
5	validation code 31					
	5.1	validate includes	32			
	5.2	is_acceptable				
	5.3	validate_addr				

6	auxi	liary functions	36		
	6.1	aux includes	37		
	6.2	xmalloc	38		
	6.3	closeall function	38		
	6.4	daemon	38		
	6.5	${\tt sighandler} \ function \ \dots $	39		
7	mai	n function	40		
8	GNU Free Documentation License				
	8.1	Applicability and Definitions	46		
	8.2	Verbatim Copying	47		
	8.3	Copying in Quantity	47		
	8.4	Modifications	48		
	8.5	Combining Documents	50		
	8.6	Collections of Documents	50		
	8.7	Aggregation With Independent Works	50		
	8.8	Translation	51		
	8.9	Termination	51		
	8.10	Future Revisions of This License $\ \ldots \ \ldots \ \ldots \ \ldots$	51		
9	Inde	ex.	53		
10	List	of code chunks	53		

### 1 Introduction

from filter uses the following API's: the libmilter API, the POSIX threads library, the OpenLDAP API, the iconv library, a rfc822 header parser and some other functions unshamefully lifted from the mutt  $^{\rm 1}$  source code.

# 2 Copying

#### This document

Copyright © Konstantinos Koukopoulos k.koukopoulos@di.uoa.gr Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections.

A copy of the license is included in the section entitled "GNU Free Documentation License".

### This program

Fromfilter, apart from beeing free documentation, is also free software. Each file produced contains the following notice:

2  $\langle copyright \ notice \ 2 \rangle \equiv$ 

(3 16 31 36 37a 40b)

/\*

This file is part of Fromfilter.

Copyright (c) 2002 Konstantinos Koukopoulos <k.koukopoulos@di.uoa.gr>

Fromfilter is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

Fromfilter is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with Fromfilter; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

\*/

Uses c 42.

 $<sup>^{1}</sup>$  a free email client available at  $\mathtt{http://www.mutt.org}$ 

# 3 libmilter API code

The code specific to the libmilter API is in the files filter.c and filter.h.

4

```
\langle filter.c 3a \rangle \equiv
3a
             \langle copyright \ notice \ 2 \rangle
             \langle filter.c\ includes\ 3d \rangle
             \langle cleanup function 15 \rangle
             \langle libmilter \ callbacks \ 4c \rangle
             \langle filter\ description\ 12 \rangle
3b
         ⟨filter.h 3b⟩≡
            \langle copyright \ notice \ 2 \rangle
            #ifndef FF_FILTER_H
            #define FF_FILTER_H
            ⟨filter.h includes 3c⟩
            ⟨struct PrivData declaration 13⟩;
             ⟨libmilter callback decls 4b⟩
            #endif
         Defines:
            FF_FILTER_H, never used.
             These two files are full of libmilter API code, so we naturally include
         the libmilter header file.
         \langle filter.h\ includes\ 3c \rangle \equiv
3c
                                                                                                (3b) 14⊳
            #include <libmilter/mfapi.h>
         \langle filter.c\ includes\ 3d \rangle \equiv
                                                                                                (3a) 5b⊳
3d
            #include <libmilter/mfapi.h>
```

#### 3.1 callback prototypes

The envelope callback is called with a null-terminated array argv which is guaranteed to contain the envelope from address in argv[0]. The rest are the ESMTP arguments.

```
3e \langle envelope \ declaration \ 3e \rangle \equiv (4)
sfsistat
envelope(SMFICTX *ctx, char *argv[])
```

The header callback is called for every header in the message body. headerf will be the header field name, headerv will be the field value.

```
3f \langle \text{header } declaration \ \text{3f} \rangle \equiv (4b 6c)
sfsistat
header(SMFICTX *ctx, char* headerf, char * headerv)
```

August 18, 2002 ../fromfilter.nw 5

The endofmessage callback is called after the message has been completely submitted. Any modifications to the message must be done here.

```
4a \langle endofmessage \ declaration \ 4a \rangle \equiv (4b 9a) sfsistat endofmessage(SMFICTX *ctx)

This filter is a message-oriented filter, so the callbacks we will be using
```

```
4b \langle libmilter\ callback\ decls\ 4b \rangle \equiv (3b) \langle envelope\ declaration\ 3e \rangle; \langle header\ declaration\ 3f \rangle; \langle endofmessage\ declaration\ 4a \rangle;
```

```
4c \langle libmilter\ callbacks\ 4c \rangle \equiv (3a)

\langle envelope\ function\ 4d \rangle

\langle header\ function\ 6c \rangle

\langle endofmessage\ function\ 9a \rangle
```

# 3.2 envelope

The envelope callback function is called by libmilter whenever a client issues a mail from command to sendmail. It first allocates and initializes the space that is private to this context <sup>2</sup>. It then retrieves some symbol values from sendmail (like the auth\_authen variable which contains the username of the authenticated user). Last, it queries the LDAP server for the necessary information.

<sup>&</sup>lt;sup>2</sup>because this is a message oriented filter, the context is the message

5a

We call smfi\_getsymval to retrieve the values of the daemon\_name, auth\_authen and auth\_author sendmail variables.

daemon\_name is the value of the "DaemonPortOptions Name=" suboption, in the sendmail configuration file. The DAEMON\_NAME macro should be defined in the config.h header, to be the name of the daemon whose messages we should filter. If the daemon\_name value is different from what we expected then we let the message pass. auth\_authen is the authentication entity of the client and auth-author is the entity the client has been authorized as.

```
\langle get \ sendmail \ symbol \ values \ 5a \rangle \equiv
                                                                             (4d)
         if (!((str = smfi_getsymval(ctx, "{daemon_name}"))!=NULL
             && !strcmp(str, DAEMON_NAME)))
             return cleanup(ctx, SMFIS_ACCEPT);
         if ((str = smfi_getsymval(ctx, "{auth_author}"))!=NULL)
             priv->auth_author = strdup( str );
         if ((str = smfi_getsymval(ctx, "{auth_authen}"))!=NULL)
             priv->auth_authen = strdup( str );
       \langle filter.c\ includes\ 3d \rangle + \equiv
5b
                                                                    (3a) ⊲3d 6b⊳
         #include <config.h>
          The smfi_setpriv libmilter call, sets the private memory for this
       context, so that the other callbacks for this message can use the same
       memory (using smfi_getpriv).
       \langle allocate \ and \ initialize \ private \ memory \ 5c \rangle \equiv
5c
         if ((priv = (struct PrivData *)calloc(1, sizeof(*priv))) == NULL)
             return SMFIS_TEMPFAIL;
         smfi_setpriv(ctx, priv);
```

The filter string is of the form "uid = username". query\_uid will use this to do an ldap search, filling priv with the necessary information from the search results.

```
\langle create\ filter\ and\ query\ LDAP\ server\ 6a \rangle \equiv
6a
                                                                        (4d)
         if (priv->auth_authen != NULL) {
            len = 5 + strlen(priv->auth_authen);
            if ((filter = (char *)malloc(len))==NULL){
               syslog(LOG, "query_id: malloc: %s\n",strerror(errno));
               return cleanup(ctx, SMFIS_TEMPFAIL);
            }
            (void)strlcpy(filter, "uid=", len);
            (void)strlcat(filter, priv->auth_authen, len);
            if (query_uid(filter, priv) <0)</pre>
                  return SMFIS_TEMPFAIL;
         } else return cleanup(ctx, SMFIS_TEMPFAIL);
         if (priv->auth_author != NULL && !strcmp(priv->auth_author, priv->auth_authen)){
                 /* TODO query_uid the author too */
        }else safe_free(priv->auth_author);
      Uses filter 12 and safe_free 37a.
6b
       \langle filter.c\ includes\ 3d \rangle + \equiv
                                                                (3a) ⊲5b 8b⊳
        #include <directory.h>
```

#### 3.3 header

The header callback function is called by libmilter for each header in the message body. For now, the only headers we are interested in are From and Sender or, if this message has been forwarded and the sender has retained the original headers, the Resent-From and Resent-Sender.

If we encounter a "Resent:" header it means that this message has been forwarded. Thus we are interested in the "Resent-\*" headers (like "Resent-From", "Resent-Sender" etc.). Any ADDRESS structures allocated by previous invocations of header must be freed.

If priv->resent is set then we advance the pointer to the header value by 7 places (if it has that many characters) and then check its value as if it is a normal header.

```
7b ⟨header function 6c⟩+≡ (4c) ⊲7a 8a⊳

if (priv->resent) {
    if (strlen(hdr) <=7)
        hdr += 7;
    else
        hdr += strlen(hdr);
}
```

Next we check to see if this header is of interest to us. If it's the first header of the sort that we've found, we parse it using the  $\mathtt{mutt}$  rfc822 and rfc2047 parsing routines.

```
\langle \text{header } function \ 6c \rangle + \equiv
                                                                      (4c) ⊲7b 8c⊳
8a
              if (!strncmp(hdr, "From", 4)){
                  if (!(priv->froms++)){
                      priv->from = rfc822_parse_adrlist(NULL, headerv);
                      rfc2047_decode_adrlist(priv->from);
                  }
              }else if (!strncmp(hdr, "Sender", 6)){
                  if (!(priv->senders++)){
                      priv->sender = rfc822_parse_adrlist(NULL, headerv);
                      rfc2047_decode_adrlist(priv->sender);
                  }
              }
8b
       \langle filter.c\ includes\ 3d \rangle + \equiv
                                                                      (3a) ⊲6b 9b⊳
         #include <rfc822.h>
         #include <rfc2047.h>
          Finally we tell sendmail to continue giving us headers.
       \langle \text{header } function \ 6c \rangle + \equiv
8c
                                                                           (4c) ⊲8a
              return SMFIS_CONTINUE;
         }
```

# 3.4 endofmessage

The message has been submitted and now we must make any changes necessary. Any information from the headers that we need has been put in the private space so we can freely delete all the headers and add our own after. If the sender is sending as himself we only need to add a From header. We call validate\_addr to check and sanitize the priv->from address and then write this address in a From header TODO there is the question if someone sending via an address like Postmaster should be mentioned in a Sender header.. If all goes well we cleanup after ourselves and the message has been filtered succesfully.

```
\langle endofmessage function 9a \rangle \equiv
9a
                                                                            (4c)
         ⟨endofmessage declaration 4a⟩
         {
             struct PrivData *priv = smfi_getpriv(ctx);
             ADDRESS *cur;
             char buf [256];
             ⟨delete headers 10a⟩
             if (priv->auth_author){
                /* TODO: <validate Sender/From headers>
                <write new Sender/From headers> */
             }else{
                cur = validate_addr(&(priv->from), priv);
                ⟨write new From header 11a⟩
             }
             return cleanup(ctx, SMFIS_CONTINUE);
         }
9b
       \langle filter.c\ includes\ 3d \rangle + \equiv
                                                                 (3a) ⊲8b 16a⊳
         #include <valid.h> /* for validate_addr */
```

The number of "From" headers is priv->froms. We use the libmilter smfi\_chgheader function with a last argument of NULL, which effectively deletes the requested header. The header to delete is specified by the second and third argument. The second argument is the name of the header field. The third argument is the index number of the header, i.e. if it is 1 then the first occurence of the header is deleted, if it is 2 the second and so on.

Because we allow only one From header (which we add in  $\langle write\ new\ From\ header\ 11a \rangle$ ) we must succeed in deleting all the headers.

```
\langle delete\ headers\ 10a \rangle \equiv
10a
                                                                (9a) 10b⊳
         #ifdef DEBUG
         syslog(LOG, "deleting %d %s headers\n", priv->froms, (priv->resent)?"Resent-From":"Fr
         syslog(LOG, "deleting %d %s headers\n", priv->senders, (priv->resent)?"Resent-Sender"
         if (priv->resent){
            while (priv->froms--)
               if (smfi_chgheader(ctx, "Resent-From", priv->froms+1, NULL) == MI_FAILURE) {;
                 syslog(LOG, "endofmessage: smfi_chgheader returned MI_FAILURE\n");
                 return cleanup(ctx, SMFIS_TEMPFAIL);
               }
         }else {
            while (priv->froms--)
               if (smfi_chgheader(ctx, "From", priv->froms+1, NULL) == MI_FAILURE) {;
                 syslog(LOG, "endofmessage: smfi_chgheader returned MI_FAILURE\n");
                 return cleanup(ctx, SMFIS_TEMPFAIL);
               }
         }
         The same goes for the Sender header:
10b
       \langle delete\ headers\ 10a\rangle + \equiv
                                                                (9a) ⊲10a
         if (priv->resent){
            while (priv->senders--)
               if (smfi_chgheader(ctx, "Resent-Sender", priv->senders+1, NULL) == MI_FAILURE)
                 syslog(LOG, "endofmessage: smfi_chgheader returned MI_FAILURE\n");
                 return cleanup(ctx, SMFIS_TEMPFAIL);
               }
         }else {
            while (priv->senders--)
               if (smfi_chgheader(ctx, "Sender", priv->senders+1, NULL) == MI_FAILURE) {;
                 syslog(LOG, "endofmessage: smfi_chgheader returned MI_FAILURE\n");
                 return cleanup(ctx, SMFIS_TEMPFAIL);
               }
         }
```

```
This is pretty self-explanatory <sup>3</sup>
        \langle write\ new\ From\ header\ 11a \rangle \equiv
11a
                                                                            (9a)
          buf[0]='\0';
          rfc822_write_address(buf, sizeof(buf), cur);
          #ifdef DEBUG
          syslog(LOG, "Adding header From: %s\n", buf);
          #endif
          if (priv->resent)
             smfi_addheader(ctx, "Resent-From", buf);
          else
             smfi_addheader(ctx, "From", buf);
        \langle write\ new\ Sender\ header\ 11b \rangle \equiv
11b
          buf[0]='\0';
          rfc822_write_address(buf, sizeof(buf), cur);
          #ifdef DEBUG
          syslog(LOG, "Adding header From: %s\n", buf);
          #endif
          if (priv->resent)
             smfi_addheader(ctx, "Resent-Sender", buf);
          else
             smfi_addheader(ctx, "Sender", buf);
```

 $<sup>^3\</sup>mathrm{Many}$  thanks go to the  $\mathtt{mutt}$  coders for these nice functions :-)

# 3.5 filter description (struct smfiDesc)

We store our filter description in the filter global variable. Our filter will modify and add headers to the message so we must set the flags member to SMFIF\_CHGHDRS|SMFIF\_ADDHDRS. For documentation on the rest of the callbacks check the libmilter documentation  $^4$ .

```
12
      \langle filter\ description\ 12 \rangle \equiv
                                                                           (3a)
        struct smfiDesc filter =
                                              /* name */
            "test filter",
            SMFI_VERSION,
                                              /* version */
            SMFIF_CHGHDRS|SMFIF_ADDHDRS, /* flags */
        /* callbacks */
            NULL,
                                              /* connect */
            NULL,
                                              /* helo */
            envelope,
                                              /* envfrom */
            NULL,
                                              /* envrcpt */
                                              /* header */
            header,
            NULL,
                                              /* eoh */
            NULL,
                                              /* body */
            endofmessage,
                                              /* eom */
            NULL,
                                              /* abort */
            NULL
                                              /* close */
        };
      Defines:
        filter, used in chunks 4d, 6a, 16c, 17a, 20-22, 31c, 32a, 40c, 41a, and 45.
```

 $<sup>^4</sup> http://sendmail.com/partner/resources/development/milter\_api/$ 

#### 3.6 struct PrivData declaration

We declare a structure PrivData that will be contained in some threadprivate memory we will allocate in (envelope function 4d). This structure must preserve, between callbacks, some values that pertain to the specific message. These are:

mail: a string that contains the attribute mail from the entry that was given by auth\_authen.

cn: a string that contains the attribute cn from the entry that was given by auth\_authen. This roughly corresponds to the Real Name of an address.

alternates: this null-terminated array of string contains the values of the multi-value attribute mailAlternateAddress. These are the username/host combinations that the user specified by auth\_authen is allowed to use in outgoing mail.

authorized: this null-terminated array of strings contains the values of the multi-value attribute mailAuthorizedAddress. These are the username/host combinations that the user specified by auth\_authen is allowed to send as.

resent: a flag that signals that the message has been forwarded and the forwarders headers are contained in from and sender.

#### from:

sender: these two structures are returned from the rfc822\_parse routines, and contain a parsed form of the values of the From and Sender headers (or the Resent-From and Resent-Sender headers if resent=1.

#### froms:

**senders:** these two integers count the number of occurences of the From and Sender headers respectively.

```
13 \langle \text{struct PrivData } declaration \ 13 \rangle \equiv  (3b)
```

```
struct PrivData {
   char *mail;
   char *cn;
   char *cn_el;
   char *auth_authen;
   char *auth_author;

   char **alternates;
   char **authorized;

   char resent;

ADDRESS *from;
ADDRESS *sender;
```

August 18, 2002 ../fromfilter.nw 15

int froms;
int senders;
};

14  $\langle filter.h\ includes\ 3c \rangle + \equiv$  (3b)  $\triangleleft 3c$  #include  $\langle rfc822.h \rangle$ 

# 3.7 cleanup

Uses safe\_free 37a.

**cleanup** frees anything that can be freed in the private structure returning the status value in 'rc'.

```
\langle cleanup function 15 \rangle \equiv
15
                                                                   (3a)
        sfsistat
        cleanup(SMFICTX *ctx, sfsistat rc)
           struct PrivData *priv = smfi_getpriv(ctx);
           int i;
           if (priv) {
                                      safe_free(priv->mail);
              if (priv->mail)
              if (priv->cn)
                                      safe_free(priv->cn);
              if (priv->auth_author) safe_free(priv->auth_author);
              if (priv->authorized){
                  for(i=0 ; priv->authorized[i] ; i++)
                      safe_free(priv->authorized[i]);
                  safe_free(priv->authorized);
              }
              if (priv->alternates){
                  for(i=0 ; priv->alternates[i] ; i++)
                      safe_free(priv->alternates[i]);
                  safe_free(priv->alternates);
              }
              if (priv->from) rfc822_free_address(&(priv->from));
              if (priv->sender) rfc822_free_address(&(priv->from));
              safe_free(priv);
              smfi_setpriv(ctx, NULL);
           }
           return rc;
       }
```

#### 3.8 includes

```
16a ⟨filter.c includes 3d⟩+≡ (3a) ⊲9b

#include <stdlib.h>

#include <syslog.h>

#include <errno.h>

#include <string.h> /* for strlcat, strlcpy etc.. */

#include <aux.h> /* for safe_free */

Uses safe_free 37a.
```

# 4 LDAP code

#### 4.1 outline

```
16b
            \langle directory.c \ 16b \rangle \equiv
               \langle copyright\ notice\ 2 \rangle
               \langle ldap \ includes \ 17a \rangle
               \langle ldap\ globals\ 18a \rangle
               \langle attribute\ handlers\ 23 \rangle
               \langle ldap \ types \ 17b \rangle
               \langle init\_ldap \ function \ 19 \rangle
               ⟨query_uid function 20b⟩
            \langle directory.h \ 16c \rangle \equiv
16c
               \langle copyright\ notice\ 2 \rangle
               #ifndef FF_LDAP_H
               #define FF_LDAP_H
               #include <filter.h>
               ⟨init_ldap declaration 18c⟩;
               \langle query\_uid \ declaration \ 20a \rangle;
               #endif
           Defines:
               FF_LDAP_H, never used.
           Uses filter 12.
```

# 4.2 Idap includes

```
\langle ldap \ includes \ 17a \rangle \equiv
                                                                        (16b)
17a
         #include <ldap.h>
         #include <pthread.h>
         #include <stdlib.h>
         #include <malloc.h>
         #include <string.h>
         #include <syslog.h>
         #include <stdio.h>
         #include <errno.h>
         #include <config.h>
         #include <filter.h>
         #include <aux.h>
       Uses filter 12.
       4.3
              types
       \langle ldap \ types \ 17b \rangle \equiv
17b
                                                                        (16b)
         typedef struct _attr_pair {
             char *name;
             int (*action)(struct PrivData *priv, LDAPMessage *p, char *attr);
         } attrib_pair;
         char * attribute_names[6] = {
             "cn; lang-el",
             "cn",
             "mailAlternateAddress",
             "mailAuthorizedAddress",
             "mail",
             NULL
         };
         attrib_pair attributes[6] = {
             { "cn; lang-el", handleCn },
             { "cn", handleCn },
             { "mailAlternateAddress", handleAlternates },
             { "mailAuthorizedAddress", handleAuthorized },
             { "mail", handleMail },
             { NULL, NULL }
         };
       Defines:
         attrib_pair, used in chunk 20b.
         attribute_names, used in chunk 21.
```

18a

(16b)

# 4.4 globals

 $\langle ldap \ globals \ 18a \rangle \equiv$ 

Uses password 42 and sizelimit 42.

This filter only makes one connection to the LDAP server, thus there is only one handle. The LDAP handle 1d is protected from concurrent accesses by the mutex variable 1d\_mutex.

```
\langle global\ ldap\ handle\ 18b \rangle
           char *search_dn;
           char *bind_dn;
           char *pass;
           int szlimit;
           struct timeval timeout;
           char *server;
        Defines:
           bind_dn, used in chunks 19, 21, 41, and 43.
           pass, used in chunks 19 and 21.
           search_dn, used in chunks 19, 21, 41, and 43.
           server, used in chunks 19, 21, 41, 43, and 44.
           szlimit, used in chunks 19 and 21.
           timeout, used in chunks 19 and 21.
18b
         \langle global\ ldap\ handle\ 18b \rangle \equiv
                                                                                     (18a)
           pthread_mutex_t ld_mutex;
           LDAP *ld;
        Defines:
           1d, used in chunks 19, 21-23, 26, 27a, 30, and 31a.
        4.5
                init_ldap
         \langle init\_ldap \ declaration \ 18c \rangle \equiv
18c
                                                                                   (16c 19)
           init_ldap( char *srv, char *binddn, char *searchdn, char *password, int sizelimit)
```

```
\langle init\_ldap \ function \ 19 \rangle \equiv
19
                                                                   (16b)
        ⟨init_ldap declaration 18c⟩
        {
           int r;
           bind_dn = BIND_DN;
           search_dn = SEARCH_DN;
           server = srv;
           szlimit = sizelimit;
           timeout.tv_sec = 10;
           timeout.tv_usec = 0;
           if (password)
              pass = strdup(password);
           else return -1;
           if (searchdn)
              search_dn = searchdn;
           if (binddn)
              bind_dn = binddn;
           if (pthread_mutex_init(&ld_mutex, NULL)){
              fprintf(stderr, "pthread_mutex_init returned 0!\n");
              return -1;
           }
           ld = ldap_init(srv, LDAP_PORT);
           if (!ld){
              perror("ldap_init");
              return -1;
           }
           ldap_set_option(ld, LDAP_OPT_SIZELIMIT, (void *)&sizelimit);
           ldap_set_option(ld, LDAP_OPT_NETWORK_TIMEOUT, (void *)&timeout);
           r = ldap_bind_s(ld, bind_dn, password, LDAP_AUTH_SIMPLE);
           if (r != LDAP_SUCCESS){
              ldap_perror(ld, "ldap_bind_s");
              return -1;
           }
           pthread_mutex_unlock(&ld_mutex);
           return 0;
        }
```

Uses bind\_dn 18a 42, 1d 18b, pass 18a, password 42, search\_dn 18a 42, server 18a 42, sizelimit 42, szlimit 18a, and timeout 18a.

```
4.6 query_uid
```

```
\langle query\_uid \ declaration \ 20a \rangle \equiv
20a
                                                                                         (16c 20b)
            query_uid(char *filter, struct PrivData *priv)
         Uses filter 12.
          \langle query\_uid function 20b \rangle \equiv
20b
                                                                                             (16b)
             \langle query\_uid \ declaration \ 20a \rangle
                 char *attr;
                 int ret=0;
                 int r;
                 LDAPMessage *p;
                 LDAPMessage *res=NULL;
                 attrib_pair *cur=NULL;
                 BerElement *berptr=NULL;
                 pthread_mutex_lock(&ld_mutex);
                 \langle do \ ldap \ search \ 21 \rangle
                 \langle handle\ ldap\ results\ 22a \rangle
                 \langle \mathit{free}\ \mathit{allocated}\ \mathit{memory}\ 22b \rangle
            done:
                 pthread_mutex_unlock(&ld_mutex);
                 return ret;
            }
          Uses attrib_pair 17b.
```

The basic search functionality is performed by calling ldap\_search\_s. The scope argument is set to LDAP\_SCOPE\_ONELEVEL; this means that we wish to search only the immediate children of the base object ( search\_dn in our case ). If the server is down, we try to reconnect.

```
\langle do \ ldap \ search \ 21 \rangle \equiv
21
                                                                  (20b)
       r=!(LDAP_SUCCESS);
       while (r != LDAP_SUCCESS){
           r = ldap_search_s(ld, search_dn,
                              LDAP_SCOPE_ONELEVEL,
                              filter, (char **)attribute_names, 0, &res);
           if (r == LDAP_SERVER_DOWN) {
              ldap_unbind_s(ld);
              ld = ldap_init(server, LDAP_PORT);
              if (!ld){
                 syslog(LOG, "query_uid: ldap_init: %s\n", strerror(errno));
                 ret = -1;
                 goto done;
              }
              ldap_set_option(ld, LDAP_OPT_SIZELIMIT, (void *)&szlimit);
              ldap_set_option(ld, LDAP_OPT_NETWORK_TIMEOUT, (void *)&timeout);
              ret = ldap_bind_s(ld, bind_dn, pass, LDAP_AUTH_SIMPLE);
              if (ret != LDAP_SUCCESS){
                 syslog(LOG, "query_uid: ldap_bind_s: %s\n", ldap_err2string(ret));
                 ret = -1;
                 goto done;
           }else if (r != LDAP_SUCCESS){
              syslog(LOG, "query_id: ldap_search_s: %s\n", ldap_result2error(ld, res, 1));
              safe_free(filter);
              ret = -1;
              goto done;
           }
       }
```

Uses attribute\_names 17b, bind\_dn 18a 42, filter 12, ld 18b, pass 18a, safe\_free 37a, search\_dn 18a 42, server 18a 42, szlimit 18a, and timeout 18a.

In \(\precase\) main function 41a\(\righta\) we have set the result number to one because we know that each user is unique. Thus we only check the first entries attributes by calling ldap\_first\_entry and then looping over the attributes.

```
\langle handle\ ldap\ results\ 22a \rangle \equiv
                                                                        (20b)
22a
         if ((p = ldap_first_entry(ld, res)) == NULL){
             syslog(LOG, "query_id: ldap_first_entry failed!\n");
             if (res) ldap_msgfree(res);
             safe_free(filter);
             ret = -1;
             goto done;
         }
         attr = ldap_first_attribute(ld, p, &berptr);
         if (attr) do {
             cur = attributes;
             while (cur->name) {
                if (!strcmp(attr, cur->name)) break;
                cur++;
             }
             if (!cur) {
                syslog(LOG, "query_id: didn't ask for this attribute: %s\n", attr);
                cur->action(priv, p, attr);
         }while ((attr = ldap_next_attribute(ld, p, berptr)) != NULL);
       Uses filter 12, ld 18b, and safe_free 37a.
       \langle free\ allocated\ memory\ 22b \rangle \equiv
22b
                                                                        (20b)
         if (!berptr) ber_free(berptr, 0);
         if (!res) ldap_msgfree(res);
         if (!p) ldap_msgfree(p);
         safe_free(filter);
       Uses filter 12 and safe_free 37a.
```

#### 4.7 attribute handlers

```
\langle attribute\ handlers\ 23 \rangle \equiv
                                                                  (16b)
23
        int
       handleAuthorized(struct PrivData *priv, LDAPMessage *p, char *attr)
           char **values;
           int num, i;
           values = ldap_get_values(ld, p, attr);
           if (values){
              num = ldap_count_values(values);
              /* mailAuthorizedAddress is a multivalue attribute, so we
               * allocate some memory for the array of values */
              if ((priv->authorized = (char **)malloc(num+1))!=NULL){
                 /* <copy [[num]] values from [[values]] to [[priv->authorized]]>> */
                 COPYMULTIVAL(values, priv->authorized);
              }else syslog(LOG, "query_id: malloc: %s\n", strerror(errno));
              ldap_value_free(values);
              return 0;
           }else
              syslog(LOG, "query_id: ldap_get_values returned null for %s: %s\n",
                 attr, ldap_err2string(ldap_result2error(ld, p, 0)));
              return -1;
           }
       }
       handleAlternates(struct PrivData *priv, LDAPMessage *p, char *attr)
        {
           int num,i;
           char **values;
           values = ldap_get_values(ld, p, attr);
           if (values){
              num = ldap_count_values(values);
```

```
/* mailAlternateAddress is a multivalue attribute, so we
       * allocate some memory for the array of values */
      if ((priv->alternates = (char **)malloc((num+1)*sizeof(char *)))!=NULL){
         /* <copy [[num]] values from [[values]] to [[priv->alternates]] test>> */
         COPYMULTIVAL(values, priv->alternates);
      }else syslog(LOG, "query_id: malloc: %s\n", strerror(errno));
      ldap_value_free(values);
      values = NULL;
      return 0;
   }else
      syslog(LOG, "query_id: ldap_get_values returned null for %s: %s\n",
         attr, ldap_err2string(ldap_result2error(ld, p, 0)));
      return -1;
   }
}
handleMail(struct PrivData *priv, LDAPMessage *p, char *attr)
   char **values;
   values = ldap_get_values(ld, p, attr);
   if (values){
      /* mail is not multivalue so we just copy it */
      if (*values)
         if ((priv->mail = strdup(*values)) == NULL)
            syslog(LOG, "query_id: strdup: %s\n", strerror(errno));
      ldap_value_free(values);
      values = NULL;
      return 0;
      syslog(LOG, "query_id: ldap_get_values returned null for %s: %s\n",
         attr, ldap_err2string(ldap_result2error(ld, p, 0)));
      return -1;
   }
}
int
handleCn(struct PrivData *priv, LDAPMessage *p, char *attr)
```

```
char **values;
 #ifdef DEBUG
     syslog(LOG, "ldap_get_values(ld, p, %s)\n", attr);
     syslog(LOG, "is %d\n", ldap_get_values(ld, p, attr));
     values = ldap_get_values(ld, p, attr);
     if (values){
        if (*values) {
           if (strlen(attr) >2) {
              if ((priv->cn_el = strdup(*values)) == NULL)
                 syslog(LOG, "query_id: strdup: %s\n", strerror(errno));
           }else{
              if ((priv->cn = strdup(*values)) == NULL)
                 syslog(LOG, "query_id: strdup: %s\n", strerror(errno));
           }
  #ifdef DEBUG
           syslog(LOG, "got %s = %s\n", attr, *values);
  #endif
        ldap_value_free(values);
        values = NULL;
       return 0;
     }else
        syslog(LOG, "query_id: ldap_get_values returned null for %s: %s\n",
           attr, ldap_err2string(ldap_result2error(ld, p, 0)));
        return -1;
     }
Uses COPYMULTIVAL 28b 37a and 1d 18b.
```

#### 4.8 foo

Uses 1d 18b.

mailAuthorizedAddress is a multivalue attribute that contains email addresses that the user can use in his body headers. We store them in priv->authorized.

```
values = ldap_get_values(ld, p, attr);

if (values){

   num = ldap_count_values(values);

   /* mailAuthorizedAddress is a multivalue attribute, so we
       * allocate some memory for the array of values */
   if ((priv->authorized = (char **)malloc(num+1))!=NULL){

            ⟨copy num values from values to priv->authorized 27b⟩

        }else syslog(LOG, "query_id: malloc: %s\n", strerror(errno));

        ldap_value_free(values);

}else syslog(LOG, "query_id: ldap_get_values returned null for %s: %s\n",
            attr, ldap_err2string(ldap_result2error(ld, p, 0)));
```

```
mailAlternateAddress is only conceptually different from mailAuthorizedAddress.
       It contains addresses that correspond to the user in some way, while
       mailAthorizedAddress contains addresses that correspond to functions
       that user may perform (like "Postmaster", "webmaster" etc..).
       \langle handle \; {\tt mailAlternateAddress} \; attribute \; 27a \rangle \equiv
27a
          values = ldap_get_values(ld, p, attr);
          if (values){
             num = ldap_count_values(values);
             /* mailAlternateAddress is a multivalue attribute, so we
              * allocate some memory for the array of values */ \,
             if ((priv->alternates = (char **)malloc((num+1)*sizeof(char *)))!=NULL){
                 /* <copy [[num]] values from [[values]] to [[priv->alternates]] test>> */
                 COPYMULTIVAL(values, priv->alternates);
             }else syslog(LOG, "query_id: malloc: %s\n", strerror(errno));
             ldap_value_free(values);
             values = NULL;
                   syslog(LOG, "query_id: ldap_get_values returned null for %s: %s\n",
         }else
                     attr, ldap_err2string(ldap_result2error(ld, p, 0)));
       Uses COPYMULTIVAL 28b 37a and 1d 18b.
27b
       \langle copy \text{ num } values \text{ } from \text{ values } to \text{ } priv-\text{>authorized } 27b \rangle \equiv
                                                                            (26)
         #ifdef DEBUG
          syslog(LOG, "printing %s values:\n", attr);
         for (i=0; i<num; i++)
             syslog(LOG, "%s\n", values[i]);
         #endif
         COPYMULTIVAL(values, priv->authorized);
       Uses COPYMULTIVAL 28b 37a.
```

```
28a
        \langle copy \; {\tt num} \; values \; from \; {\tt values} \; to \; {\tt priv->alternates} \; test \; 28a 
angle \equiv
          #ifdef DEBUG
          syslog(LOG, "printing %s values:\n", attr);
          for (i=0; i<num; i++)</pre>
              syslog(LOG, "%s\n", values[i]);
          #endif
          COPYMULTIVAL(values, priv->alternates);
        Uses COPYMULTIVAL 28b 37a.
        \langle defines 28b \rangle \equiv
28b
          #define COPYMULTIVAL(A,B) \
          {for (i=0; i<num; i++)\
              if ((B[i] = strdup(A[i])) == NULL){\
                  syslog(LOG, "query_id: strdup: %s\n", strerror(errno));\
                  break;\
              }\
          B[i]=NULL;}
        Defines:
          COPYMULTIVAL, used in chunks 23, 27, and 28a.
```

```
29
       \langle copy \; {\tt num} \; values \; from \; {\tt values} \; to \; {\tt priv->alternates} \; {\tt 29} \rangle \equiv
         #ifdef DEBUG
         syslog(LOG, "printing %s values:\n", attr);
         for (i=0; i<num; i++)</pre>
             syslog(LOG, "%s\n", values[i]);
         #endif
         for (i=0; i<num; i++)
             if ((priv->alternates[i] = strdup(values[i])) == NULL){
                syslog(LOG, "query_id: strdup: %s\n", strerror(errno));
                break;
         priv->alternates[i]=NULL;
```

cn isn't multivalue so things are simpler  $\langle handle \ {\tt cn} \ attribute \ 30 \rangle \equiv$ 30 #ifdef DEBUG syslog(LOG, "ldap\_get\_values(ld, p, %s)\n", attr); syslog(LOG, "is %d\n", ldap\_get\_values(ld, p, attr)); #endif values = ldap\_get\_values(ld, p, attr); if (values){ if (\*values) { if (strlen(attr) >2) { if ((priv->cn\_el = strdup(\*values)) == NULL) syslog(LOG, "query\_id: strdup: %s\n", strerror(errno)); }else{ if ((priv->cn = strdup(\*values)) == NULL) syslog(LOG, "query\_id: strdup: %s\n", strerror(errno)); } #ifdef DEBUG syslog(LOG, "got %s = %s\n", attr, \*values); #endif } ldap\_value\_free(values); values = NULL; }else syslog(LOG, "query\_id: ldap\_get\_values returned null for %s: %s\n", attr, ldap\_err2string(ldap\_result2error(ld, p, 0)));

Uses 1d 18b.

August 18, 2002

# 5 validation code

```
31b \langle valid.c \ 31b \rangle \equiv \langle copyright \ notice \ 2 \rangle \\ \langle validate \ includes \ 32a \rangle \\ \langle is\_acceptable \ function \ 32b \rangle \\ \langle validate\_addr \ function \ 33b \rangle

31c \langle valid.h \ 31c \rangle \equiv \langle copyright \ notice \ 2 \rangle \\ \#include \ \langle filter.h \rangle /* \ for \ struct \ PrivData \ structure \ */ \langle validate\_addr \ declaration \ 33a \rangle;

Uses filter 12.
```

}

#### 5.1 validate includes

```
\langle validate\ includes\ 32a \rangle \equiv
                                                                         (31b)
32a
         #include <stdio.h>
         #include <stdlib.h>
         #include <string.h>
         #include <ctype.h>
         #include <syslog.h>
         #include <config.h>
         #include <filter.h>
         #include <rfc822.h>
         #include <iconv.h>
         #include <aux.h>
       Uses filter 12.
       5.2
              is_acceptable
       is_acceptable checks str against one and the members of alters, re-
       turning 1 if the match was succesfull.
       \langle is\_acceptable function 32b \rangle \equiv
32b
                                                                        (31b)
         is_acceptable(char *one, char** alters, char *str)
             int i;
         #ifdef DEBUG
                syslog(LOG, "Checking if \"%s\" is acceptable\n", str);
         #endif
             if (one && !strcmp(one, str)) return 1;
             if (alters ) {
                for (i = 0; alters[i]; i++) {
                     if (!strcmp(alters[i], str)) return 1;
             }
             return 0;
```

#### 5.3 validate\_addr

The priv->from variable contains an ADDRESS structure. This structure is a linked list of all the addr objects from the addrlst object contained in priv->from. We must call is\_acceptable (see \( is\_acceptable \) function 32b\( ) at least once for every addr object that has a mailbox. If the mailbox isn't acceptable with priv->alternates then it must be acceptable with priv->authorized.

If one of the addr objects is a group object then from that object on a sublist exists (terminated by a null object).

If no valid mailbox is found, then we create our own ADDRESS structure with the mailbox member set to the LDAP attribute priv->mail. Also we decide if the personal member will be priv->cn or priv->cn;lang-el (unimplemented - currently we set personal to priv->cn in a very fascist way:-)

```
⟨validate_addr declaration 33a⟩≡
                                                                           (31c 33b)
33a
          ADDRESS*
          validate_addr(ADDRESS **addr, struct PrivData* priv)
        \langle validate\_addr function 33b \rangle \equiv
33b
                                                                              (31b)
           ⟨validate_addr declaration 33a⟩
              ADDRESS *cur;
              int authorized=0;
              iconv_t cd;
              char buf[BUFSIZ];
              char *tobuf;
              char *frombuf;
              char *from_code;
              int len, fleft, tleft;
              for (cur = *addr; cur != NULL; cur=cur->next){
                  ⟨if *cur valid break 34a⟩
              if (cur == NULL) {
                  \langle make \ new \ ADDRESS \ 34b \rangle
              }
              \langle check\ cur-\dot{\epsilon}personal\ 35 \rangle
              if (!cur->personal){
                  cur->personal = (char *)strdup(priv->cn);
              return cur;
          }
```

We use the <code>is\_acceptable</code> function to check <code>cur->mailbox</code> against <code>priv->mail</code>, <code>priv->alternates</code> and <code>priv->authorized</code>. If <code>cur</code> is a group item then we advance to the first item in the group.

```
\langle if * cur \ valid \ break \ 34a \rangle \equiv
34a
                                                                          (33b)
          if (cur->group && cur->next)
             cur=cur->next;
          if (cur->mailbox && is_acceptable(priv->mail, priv->alternates, cur->mailbox)) break;
          if (cur->mailbox && is_acceptable(NULL,
                                                               priv->authorized, cur->mailbox)) {
             authorized=1;
             break;
         }
34b
        \langle make \ new \ ADDRESS \ 34b \rangle \equiv
                                                                          (33b)
          cur = (ADDRESS*) calloc(sizeof(ADDRESS), 1);
          if (priv->mail) cur->mailbox = strdup(priv->mail);
          rfc822_free_address(addr);
          *addr = cur;
```

```
\langle check\ cur-\dot{c}personal\ 35\rangle \equiv
35
                                                                  (33b)
        if (cur->personal && !strncmp(cur->personal, "=?", 2)){
           len = strpbrk(cur->personal+2, "?") - cur->personal - 1;
           from_code = (char *)malloc(len);
           (void)strlcpy(from_code, cur->personal+2, len);
           frombuf = from_code;
           while (*(frombuf)!='\setminus0') {
              *frombuf = toupper(*(frombuf));
              frombuf++;
           }
           cd = iconv_open("UTF-8", (const char *)from_code);
           if (cd != (iconv_t)-1) {
              fleft = strlen(cur->personal_decoded);
              frombuf = cur->personal_decoded;
              tleft = BUFSIZ;
              tobuf = buf;
              len = iconv(cd, (const char **) &frombuf, &fleft, &tobuf, &tleft);
              if (fleft == 0 && len != -1) {
                 if (memcmp(priv->cn_el, buf, strlen(priv->cn_el))){
                    safe_free(cur->personal);
                 }
              }else{
                 syslog(LOG, "iconv: %s\n", strerror(errno));
                 safe_free(cur->personal);
              }
              (void)iconv_close(cd);
           }else{
              syslog(LOG, "iconv_open(UTF-8, %s) failed: %s\n", from_code, strerror(errno));
              safe_free(cur->personal);
           }
       }else{
           safe_free(cur->personal);
        /*
       #ifdef DEBUG
        syslog(LOG, "encoding is %s, length %d chars\n", cur->personal+2, strpbrk(cur->person
        #endif
```

```
if (!strncmp(cur->personal+2, "utf-8", strpbrk(cur->personal+2, "?") - cur->person
        if (priv->cn_el){
  #ifdef DEBUG
  syslog(LOG, "1: %s\n", cur->personal_decoded);
  syslog(LOG, "2: %s\n", priv->cn_el);
 #endif
           if (memcmp(cur->personal_decoded, priv->cn_el, strlen(priv->cn_el))){
              safe_free(cur->personal);
           };
       }else{
            TODO convert priv->cn to unicode so we can compare with cur->personal_decod
        }
 #if 0
     }else if (..) {
        do this for every encoding we support
 #endif
     }else{
        safe_free(cur->personal);
Uses safe_free 37a.
```

## 6 auxiliary functions

```
\begin{array}{ll} 36 & \langle aux.c \ 36 \rangle \equiv \\ & \langle copyright \ notice \ 2 \rangle \\ & \langle aux \ includes \ 37b \rangle \\ & \langle xmalloc \ function \ 38b \rangle \\ & \langle closeall \ function \ 38c \rangle \\ & \langle daemon \ function \ 39a \rangle \\ & \langle sighandler \ function \ 40a \rangle \end{array}
```

August 18, 2002

```
\langle aux.h \ 37a \rangle \equiv
37a
           \langle copyright \ notice \ 2 \rangle
           #ifndef FF_AUX_H
           #define FF_AUX_H
           #define safe_free(A) {free(A); (A)=NULL;}
           #define COPYMULTIVAL(A,B) \
           {for (i=0; i<num; i++)\
               if ((B[i] = strdup(A[i])) == NULL)\{\setminus
                   syslog(LOG, "query_id: strdup: %s\n", strerror(errno));\
                   break; \
               }B[i]=NULL;}
           \langle xmalloc \ declaration \ 38a \rangle;
           \langle daemon \ declaration \ 38d \rangle;
           \langle sighandler \ declaration \ 39b \rangle;
           #endif
        Defines:
           COPYMULTIVAL, used in chunks 23, 27, and 28a.
           FF\_AUX\_H, never used.
           safe_free, used in chunks 6a, 15, 16a, 21, 22, and 35.
        6.1
               aux includes
         \langle aux \ includes \ 37b \rangle \equiv
37b
                                                                                     (36)
           #include <stdio.h>
           #include <stdlib.h>
           #include <string.h>
           #include <unistd.h>
           #include <syslog.h>
           #include <errno.h>
           #include <signal.h>
           #include <ctype.h>
```

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#include <config.h>

#### 6.2xmalloc

```
The common safe {\tt malloc} replacement:
         \langle xmalloc \ declaration \ 38a \rangle \equiv
38a
                                                                                       (37a 38b)
            void *
            xmalloc(int sz)
         \langle \mathtt{xmalloc}\ \mathit{function}\ \mathtt{38b} \rangle \equiv
38b
                                                                                            (36)
            \langle \mathtt{xmalloc}\ declaration\ 38a \rangle
                void *ptr = NULL;
                ptr = malloc(sz);
               if (ptr == NULL) {
                   syslog(LOG, "xmalloc: %s", strerror(errno));
                   exit(EXIT_FAILURE);
               }
               return ptr;
            }
                 closeall function
         6.3
         \langle closeall \ function \ 38c \rangle \equiv
38c
                                                                                            (36)
            void
            closeall(int fd)
                  int fdlimit = sysconf(_SC_OPEN_MAX);
                  while (fd < fdlimit)</pre>
                    close(fd++);
            }
         6.4
                 daemon
         \langle daemon \ declaration \ 38d \rangle \equiv
38d
                                                                                       (37a 39a)
```

daemon(int nochdir, int noclose)

```
August 18, 2002
```

```
../fromfilter.nw 40
```

```
\langle daemon \ function \ 39a \rangle \equiv
39a
                                                                           (36)
          \langle daemon \ declaration \ 38d \rangle
              switch (fork())
                   case 0: break;
                   case -1: return -1;
                   default: _exit(0);
                                                   /* exit the original process */
              if (setsid() < 0)
                                                   /* shoudn't fail */
                 return -1;
              /* dyke out this switch if you want to acquire a control tty in */
              /* the future -- not normally advisable for daemons */
              switch (fork())
                   case 0: break;
                   case -1: return -1;
                   default: _exit(0);
              }
              if (!nochdir)
                 chdir("/");
              if (!noclose)
                   closeall(0);
                   open("/dev/null",0_RDWR);
                   dup(0); dup(0);
              }
              return 0;
          }
              sighandler function
        \langle sighandler \ declaration \ 39b \rangle \equiv
                                                                       (37a 40a)
39b
```

sighandler(int signum)

```
\langle \mathtt{sighandler} \; \mathit{function} \; 40a \rangle \equiv
40a
                                                                                               (36)
            \langle sighandler \ declaration \ 39b \rangle
            syslog(LOG, "Got signal %d..\n", signum);
                main function
         \langle main.c \ 40b \rangle \equiv
40b
            \langle copyright \ notice \ 2 \rangle
            \langle main\ includes\ 40c \rangle
            \langle main \ function \ 41a \rangle
40c
         \langle main\ includes\ 40c \rangle \equiv
                                                                                             (40b)
            #include <stdio.h>
            #include <stdlib.h>
            #include <string.h>
            #include <unistd.h>
            #include <syslog.h>
            #include <errno.h>
            #include <signal.h>
            #include <ctype.h>
            #include <config.h>
            #include <filter.h>
```

#include <directory.h>
#include <aux.h>

Uses filter 12.

August 18, 2002

```
\langle main function 41a \rangle \equiv
                                                              (40b) 41b⊳
41a
        extern struct smfiDesc filter;
         void
        usage(void)
            printf("Usage: filter [options] -p sock\n");
           printf(" options are:\n");
           printf(" \t -p <file>
                                      \t\t unix socket to rendevouz with sendmail.\n");
           printf(" \t -b <bind_dn> \t\t DN to bind on the LDAP server\n");
            printf(" \t -s <search_dn> \t\t base DN to use for searches \n");
           printf(" \t -H <host> \t\t what host the LDAP server is running on. \n");
            printf(" \ t -P < password > \ t \ simple authentication password for LDAP
         \t\t\t server (will prompt if missing).\n");
            printf(" \t -h
                                      \t\t this message.\n");
        }
```

Uses bind\_dn 18a 42, filter 12, password 42, search\_dn 18a 42, and server 18a 42.

```
\langle main \ function \ 41a \rangle + \equiv
                                                                             (40b) ⊲41a
41b
           main(int argc, char *argv[])
               \langle local\ vars\ and\ initialization\ 42 \rangle
               while ((c = getopt(argc, argv, "db:p:H:h:s:f:P:p:")) != (char)EOF)
                   \langle handle\ flags: switch(c) 43\rangle
               }
               \langle set \ defaults \ 44 \rangle
               if (init_ldap(server, bind_dn, search_dn, password, sizelimit) <0)</pre>
                   exit(EXIT_FAILURE);
               act.sa_handler = sighandler;
               sigaction(SIGSEGV, &act, NULL);
               if (is_daemon) daemon(1,0);
               ⟨register filter and call smfi_main 45⟩
           }
         Defines:
           main, never used.
         Uses act 42, bind_dn 18a 42, c 42, is_daemon 42, password 42, search_dn 18a 42,
           server 18a 42, and sizelimit 42.
```

After declaring the local variables, we must initialize some of them so that they have sane values. server is the hostname the user chose, bind\_dn and search\_dn are the LDAP DNs, password is the LDAP simple auth password and sizelimit is the limit on LDAP entries returned from a search.

```
\langle local\ vars\ and\ initialization\ 42 \rangle \equiv
42
                                                                                 (41b)
         char c, *password, *server, *bind_dn;
         char *search_dn;
         int sizelimit;
         int is_daemon;
         struct sigaction act;
         server = NULL;
         bind_dn = NULL;
         search_dn =NULL;
         password = NULL;
         sizelimit = 1;
         is_daemon = 0;
       Defines:
         act, used in chunk 41b.
         bind_dn, used in chunks 19, 21, 41, and 43.
         c, used in chunks 2, 41b, and 43.
         is_daemon, used in chunks 41b and 43.
         password, used in chunks 18c, 19, 41, 43, and 44.
         search_dn, used in chunks 19, 21, 41, and 43.
         server, used in chunks 19, 21, 41, 43, and 44.
         sizelimit, used in chunks 18c, 19, and 41b.
```

getopt returns in c the character of each flag it encounters in the command line arguments. We do a switch on c to handle each flag. Most of these are self-explanatory, but for an explanation check (usage function (never defined)). Of interest is the 'p' option, where the connection with sendmail is set up. We only support local/unix sockets for now. The user specifies them like "unix:/var/run/f1.sock" so we must discard the leading "unix:".

```
\langle handle\ flags: switch(c)\ 43 \rangle \equiv
43
                                                                    (41b)
        switch (c)
        {
           case 'd':
              is_daemon = 1;
              break;
           case 'h':
              usage();
              exit(EXIT_SUCCESS);
           case 'H':
              if (optarg == NULL) {
                 fprintf(stderr, "missing arg\n");
                  exit(EXIT_FAILURE);
              }
              server = (char *)strdup(optarg);
              break:
           case 's':
              if (optarg == NULL ){
                 fprintf(stderr, "missing arg\n");
                  exit(EXIT_FAILURE);
              }
              search_dn = (char *)strdup(optarg);
           case 'b':
              if (optarg == NULL ){
                 fprintf(stderr, "missing arg\n");
                  exit(EXIT_FAILURE);
              }
              bind_dn = (char *)strdup(optarg);
              break;
           case 'P':
              if (optarg == NULL ){
                 fprintf(stderr, "missing arg\n");
                 exit(EXIT_FAILURE);
              }
              password = (char *)strdup(optarg);
              break;
           case 'p':
```

```
if (!(optarg && *optarg)){
                   fprintf(stderr, "Bad port\n");
                   exit(EXIT_FAILURE);
               }
               if (smfi_setconn(optarg) == MI_FAILURE)
                   (void) fputs("smfi_setconn failed\n", stderr);
                   exit(EXIT_FAILURE);
               if (!strncmp(optarg, "unix:", 5))
                  unlink(optarg + 5);
               else if (!strncmp(optarg, "local:", 6))
                  unlink(optarg + 6);
               break;
            case '?':
            default:
               usage();
               exit(EXIT_FAILURE);
        }
      Uses bind_dn 18a 42, c 42, is_daemon 42, password 42, search_dn 18a 42,
        and server 18a 42.
         If, after parsing command line options, some values are left unset we
      set the default values. password is a special case were we must query the
      user for the password.
      \langle set \ defaults \ 44 \rangle \equiv
44
                                                                        (41b)
        if (!server) {
            server = (char *)strdup(HOST);
            if (!server) {
               perror("strdup");
               exit(EXIT_FAILURE);
            }
        }
        if (!password) {
            password = getpassphrase("password for ldap server:");
            if (!password) {
               perror("getpass");
               exit(EXIT_FAILURE);
            }
        }
      Uses password 42 and server 18a 42.
```

Finaly we register our filter with the libmilter subsystem and enter the smfi\_main. If smfi\_main ever returns then surely an error has occured so we return the error code as our exit status.

### 8 GNU Free Documentation License

Version 1.1, March 2000

Copyright © 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

#### Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

#### 8.1 Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that

says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, IFTEX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

#### 8.2 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

#### 8.3 Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back

cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

#### 8.4 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- State on the Title page the name of the publisher of the Modified Version, as the publisher.
- Preserve all the copyright notices of the Document.
- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- Include an unaltered copy of this License.
- Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement

made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

#### 8.5 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

#### 8.6 Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

#### 8.7 Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

#### 8.8 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

#### 8.9 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

#### 8.10 Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

# ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

#### 9 Index

Here is a list of the identifiers used, and where they appear. Underlined entries indicate the place of definition. This index is generated automatically.

```
act: 41b, 42
                                       ld: <u>18b</u>, 19, 21, 22a, 23, 26, 27a,
attrib_pair: 17b, 20b
                                          30, 31a
attribute_names: 17b, 21
                                       main: 41b
bind_dn: 18a, 19, 21, 41a, 41b, 42,
                                       pass: <u>18a</u>, 19, 21
  43
                                       password: 18c, 19, 41a, 41b, 42,
c: 2, 41b, <u>42</u>, 43
                                          43, 44
COPYMULTIVAL: 23, 27a, 27b, 28a,
                                       safe_free: 6a, 15, 16a, 21, 22a,
  28b, 37a
                                          22b, 35, 37a
FF_AUX_H: 37a
                                       search_dn: 18a, 19, 21, 41a, 41b,
FF_FILTER_H: 3b
                                          42, 43
FF_LDAP_H: 16c
                                       server: 18a, 19, 21, 41a, 41b, 42,
filter: 4d, 6a, 12, 16c, 17a, 20a,
                                          43, 44
  21, 22a, 22b, 31c, 32a, 40c, 41a,
                                       sizelimit: 18c, 19, 41b, \underline{42}
                                       szlimit: <u>18a</u>, 19, 21
is_daemon: 41b, \underline{42}, 43
                                       timeout: 18a, 19, 21
```

#### 10 List of code chunks

This list is generated automatically. The numeral is that of the first definition of the chunk.

```
\langle cleanup function 15 \rangle
⟨closeall function 38c⟩
\langle daemon \ declaration \ 38d \rangle
(daemon function 39a)
\langle endofmessage \ declaration \ 4a \rangle
\langle endofmessage function 9a \rangle
\langle envelope \ declaration \ 3e \rangle
⟨envelope function 4d⟩
\langle header \ declaration \ 3f \rangle
⟨header function 6c⟩
⟨init_ldap declaration 18c⟩
⟨init_ldap function 19⟩
⟨is_acceptable function 32b⟩
(main function 41a)
⟨query_uid declaration 20a⟩
(query_uid function 20b)
⟨sighandler declaration 39b⟩
⟨sighandler function 40a⟩
⟨struct PrivData declaration 13⟩
⟨validate_addr declaration 33a⟩
⟨validate_addr function 33b⟩
(xmalloc declaration 38a)
\langle xmalloc function 38b \rangle
```

```
⟨libmilter callback decls 4b⟩
⟨libmilter callbacks 4c⟩
(allocate and initialize private memory 5c)
\langle attribute\ handlers\ 23 \rangle
\langle aux \ includes \ 37b \rangle
\langle aux.c \ 36 \rangle
\langle aux.h \ 37a \rangle
⟨check cur-¿personal 35⟩
\langle copy \text{ num } values \text{ } from \text{ values } to \text{ priv->alternates } 29 \rangle
\langle copy \text{ num } values \text{ } from \text{ values } to \text{ priv->alternates } test \text{ } 28a \rangle
\langle copy \text{ num } values \text{ } from \text{ values } to \text{ priv->authorized } 27b \rangle
\langle copyright\ notice\ 2 \rangle
⟨create filter and query LDAP server 6a⟩
\langle defines 28b \rangle
⟨delete headers 10a⟩
\langle directory.c \ 16b \rangle
\langle directory.h \ 16c \rangle
⟨do ldap search 21⟩
\langle filter\ description\ 12 \rangle
\langle filter.c 3a \rangle
⟨filter.c includes 3d⟩
⟨filter.h 3b⟩
⟨filter.h includes 3c⟩
(free allocated memory 22b)
⟨get sendmail symbol values 5a⟩
⟨global ldap handle 18b⟩
⟨handle cn attribute 30⟩
⟨handle mail attribute 31a⟩
\langle handle \; {\tt mailAlternateAddress} \; attribute \; 27a \rangle
\langle handle \; {\tt mailAuthorizedAddress} \; attribute \; 26 \rangle
\langle handle\ flags: switch(c) 43\rangle
⟨handle ldap results 22a⟩
⟨if *cur valid break 34a⟩
⟨ldap globals 18a⟩
(ldap includes 17a)
⟨ldap types 17b⟩
\langle local\ vars\ and\ initialization\ 42 \rangle
⟨main includes 40c⟩
\langle main.c 40b \rangle
\langle make \ new \ ADDRESS \ 34b \rangle
⟨register filter and call smfi_main 45⟩
⟨set defaults 44⟩
\langle valid.c \ 31b \rangle
⟨valid.h 31c⟩
⟨validate includes 32a⟩
⟨write new From header 11a⟩
⟨write new Sender header 11b⟩
```