Using Discuss*

The Student Information Processing Board

Stan Zanarotti
Ken Raeburn
Nancy Gilman
Richard Barbalace

Revision 1.6
June 1992

Contents

1 Introduction

1.1 What is DISCUSS?

DISCUSS is a networked electronic conferencing system. Similar to the bulletin boards that have become popular on microcomputers, it differs in its accessibility from time-sharing machines and workstations on the network. DISCUSS allows a user to attend a set of electronic meetings for discussing various topics. Each meeting has its own specific focus, such as a project, class, or special interest.

Users may select ("attend") meetings, and read messages ("transactions") that have been entered in those meetings. A user may also enter new transactions into a meeting, either by replying to a previous one or starting one on a new topic.

1.2 Clients and Servers

DISCUSS uses a client/server model of networking. A client is a program run by you, the user, to read meetings and otherwise work with DISCUSS. DISCUSS servers actually store the information kept in the meetings, while the DISCUSS client programs handle the user interface. When a user runs a DISCUSS client program, it contacts the various servers to manipulate the meetings.

Three main DISCUSS client programs are available: terminal-oriented *discuss* for running in an **xterm** window or over dialup; *edsc* for running within **emacs**; and an **X**-based client, *xdsc* which opens a new window. The use of each of these clients is described in the following sections.

Because of the networked nature of this system, some sort of authentication scheme needs to be used. If such a scheme is not used, it becomes easy to generate fake electronic mail messages and enter them as transactions. In order to prevent fake transactions and to ensure privacy, the Kerberos system, developed on MIT's Project Athena, is being used to authenticate user transactions. Although Kerberos is not completely secure, it is the most secure system now available, even if it is not the simplest. If you are using an Athena workstation, there is nothing further you need to do to use Kerberos; you can start using DISCUSS immediately. If you are not on an Athena workstation, refer to Section 8.1 for more information about using Kerberos.

1.3 Keeping track of meetings

DISCUSS does not keep a central list of all DISCUSS meetings. Instead, it maintains a list for each user of the meetings that they attend. To specify a meeting you can use the meeting name, or your own abbreviation. For example, the *Discuss_development* meeting could be known as *discuss* to one user, *dsc* to another, or by both to yet another user. All this information, as well as the information needed by the client programs, is stored in a file called **.meetings** in your home directory. Without this file, DISCUSS will not be able to locate meetings.

1.4 DISCUSS client programs

The best way to understand the ideas above is to start playing with the system. They should become clear after a little practice. Each of the following three sections describe one of the DISCUSS client programs available on Athena.[1] Each has its own advantages, and which one you decide to use is a matter of preference.

[1] Throughout this document, DISCUSS client names, such as *discuss*, are printed in italics. DISCUSS meetings also are displayed in italic print. Commands that the user should enter or keys that should be pressed are quoted in boldface. Output from the clients is shown in `typewriter` print.

Most users start by using *discuss*, a no-frills client which can be run at any time and which provides help for every command. While *xdsc* cannot be run over dialup and is more limited than *discuss*, *xdsc* provides an intuitive, pretty interphase designed to be as user-friendly as possible. The *edsc* client is convenient to those who are fond of using **emacs**; it is perhaps less easy to become accustomed to than *xdsc*, but may be run over dialup.

2 Using *discuss*

Here is a step-by-step explanation of how to use the client *discuss*, which may be run in an **xterm** window or over dialup.

2.1 Initial setup

To run Discuss in an xterm window, enter "**discuss**" at your **athena%** prompt. The first thing it will do is look for a **.meetings** file. If you do not have one it will assume this is your first time running Discuss. The client will then ask you if you want to create a **.meetings** file. Enter "**y**" for yes. Your first time using Discuss a prototype **.meetings** file will be created in your home directory for you.

Running *discuss* without a **.meetings** file looks like this:

```
athena% discuss
discuss: No such file or directory

If you are using discuss for the first time, or if you have only used the
experimental version of discuss, you need to run the 'dsc_setup'
command from the shell.

Run dsc_setup now? (y or n)  y

Running dsc_setup...
  Creating .meetings file:
  done.
discuss:
```

The version of this program distributed on Athena will give you a **.meetings** file that includes two meetings, called *New_Meetings* and *Everybody.* The former is for announcements of new meetings, and the latter is for random announcements, questions, and anything else that may be of general interest.

2.2 Adding and reading meetings

The meetings you currently attend are listed in your **.meetings** file. To list the meetings in this file enter the "**list_meetings**" command:

```
discuss:  list_meetings
 Flags  Meeting
 -----  -------
 c      New_Meetings, new_meetings
 c      Everybody, eve
discuss:
```

The "c" flag means that the meeting has changed so that it contains new unread transactions. To attend the *New_Meetings* meeting, simply "**goto**" it. You may use any of the names listed to attend a meeting.[2] Note the underscore character in the name *new_meetings*. You should see something like this:

```
discuss:  goto new_meetings
New_Meetings meeting;  433 new, 433 last.
discuss:
```

Apparently over 400 transactions have been entered in this meeting, none of which have you read yet. The transactions in *New_Meetings* can be used to add the meetings you wish to "attend" to your personal list of meetings. Perhaps the best way to find meetings that interest you is to read through all of the transactions in *New_Meetings*. We start by listing the first hundred transactions:

```
discuss:  list first,first+100
 [0001]*  (2) 10/29/86 21:54 wesommer     Reason for this meeting
 [0034]   (9) 04/10/87 23:54 rlk          Athena_Flames meeting
 [0045]  (10) 04/11/87 04:24 spook        Athena_Meetings meeting
 [0052]   (8) 04/12/87 02:35 rfrench      Stories meeting
 [0060]  (13) 04/22/87 12:29 tony         laser-lovers meeting
 [0070]  (11) 05/01/87 17:06 geer         x-conversion-meeting meeting
 [0071]   (9) 05/14/87 17:11 tony         MDQS_development meeting
 [0073]   (6) 05/17/87 03:05 rfrench      GNU_C meeting
 [0076]   (6) 05/17/87 03:07 rfrench      Printers meeting
 [0080]   (9) 05/20/87 21:20 tytso        Quotes_Misquotes_and_Out_of_Con...
 [0081]   (9) 05/21/87 02:02 rfrench      Bug_GNU_Emacs meeting
 [0093]  (11) 06/05/87 13:35 ambar        MIT_Clearing_House meeting
 [0094]  (35) 06/07/87 00:21 tytso        RISKS Forum meeting
 [0101]   (8) 06/29/87 16:20 wesommer     SMS meeting
discuss:
```

The different columns in the listing show the number of each specified "transaction" in the meeting (with a "*" next to the "current" one), the length in lines, the date and time of entry of the transaction, the user who entered the transaction, and the title given to it. Note that some numbers are unused in the sequence; these transactions have been deleted, probably by the user who put them there. They are therefore invisible to you, unless retrieved by the writers of the transactions at a later time.

Transactions are selected by strings called "transaction specifiers". Usually a transaction specifier is just a transaction number, but it can also be a range or a keyword. We could have entered "list 1:101" to get the preceding list. To specify the first ten messages you can type "1,10" or "1:10". Keywords ("current", "all", "last", "first", "next", "prev", etc.) can be used to refer to individual transactions or a range of transactions. Expressions ("last-10", "current+2") allow transaction specifiers to be combined, and also demonstrate that *discuss* can do simple arithmetic. Other transaction specifiers have to do with chaining ("lref", "fref", "pref", "nref", etc.), which is discussed below. And still others can make using Discuss a lot easier, like "new", which are the transactions at the end of a meeting that you have not seen. The *discuss* command interface

[2]The **.meetings** implementation allows for any number of names for a meeting. The names are kept in a comma-separated list in the **.meetings** file, which you can edit with **emacs** if you wish to change meetings' names.

milks transaction specifiers for all it can get, and you should too. For more information type "**help spec**" in *discuss*.

As mentioned above, looking through all of *New_Meetings* is one of the best ways to find those meetings which interest you. To learn more about a meeting listed in *New_Meetings*, "**print**" the corresponding transaction. (This transaction is usually the same as the first transaction in the announced meeting). For instance, if *Quotes_Misquotes_and_Out_of_Con...*[3] has you wondering, you would do the following:

```
discuss:  print 80
[0080] tytso@ATHENA.MIT.EDU  New_Meetings  05/20/87 21:20 (9 lines)
Subject: Quotes_Misquotes_and_Out_of_Context_Quotes meeting
  Meeting Name:  Quotes_Misquotes_and_Out_of_Context_Quotes
  Host:          BLOOM-BEACON.MIT.EDU
  Pathname:      /usr/spool/discuss/quotes
  Participation: Public

This is the reincarnation of the "quotes" discuss meeting, moved
to Bloom-Beacon, that creater of fine quality news and other sundry
information.  Please put quotes in here and leave .plan files for
OTHER slanders and libels......[4]
--[0080]--
^L
discuss:
```

The "host" and "pathname" fields are for the "**add_meeting**" routine to use to tell DISCUSS where to find the meeting. The "participation" field has no effect on the access control; it simply states whether the announcer considers the meeting to be public or private in nature.

The ^L at the end of the transaction marks the end of the message. If you are reading a range of transactions, the ^L signals the "**print**" command (which uses **more** by default) to stop and wait for you to press the space bar before going on to the next transaction. If a transaction takes up more space than can fit on a single screen, "**print**" will put up the first screen and wait for you to press the spacebar before going on to the next screen.

If you want to read the transactions in *quotes*, you need to add the meeting to your personal list of meetings with the command "**add_meeting**":

```
discuss:  add_meeting
Transaction [0080] Meeting quotes (quotes) added.
discuss:
```

With no arguments, "**add_meeting**" looks at the current transaction for a meeting announcement, and adds that meeting to your private list of meetings. If you had not printed the meeting announcement first, then you would receive an error since "**add_meeting**" would not know what meeting you wanted to attend.

In addition to adding a meeting after printing a meeting announcement in *New_Meetings*, the "**add_meeting**" command can take two types of arguments to allow for more ways of adding meetings. Entering "**add_meeting 80**" would also cause *quotes* to be added by causing the command to

[3] The ellipsis simply means the meeting title was too long to fit on a single line and has been truncated.

[4] A reference to the *Central_America* meeting, which archives all changed **.plan** files each night.

check transaction 80. Another way this meeting can be added is by specifying the full host and pathname fields with "**add_meeting bloom-beacon.mit.edu:/usr/spool/discuss/quotes**"; note the colon separating the host and pathname. This last method is useful if you do not know where the meeting is announced.

Now "**list_meetings**" (or "**lsm**" for short) will show your new addition:

```
discuss:  lsm
 Flags  Meeting
 -----  -------
 c      New_meetings, new_meetings
 c      Everybody, eve
 c      quotes, quotes
discuss:
```

2.3 Chains

Looking at the DISCUSS logo on the cover, you will notice that the letters are linked together by chains. From the above examples, you should also notice that DISCUSS numbers transactions from [0001] on. Chains group transactions by subject.

When a user enters a new transaction into a DISCUSS meeting, he or she has the choice of starting a new chain, or adding onto an existing chain. The "**talk**" command will prompt the user for a subject, and will enter the transaction as the start of a new chain. The "**reply**" command takes an already existing transaction, and adds the new transaction onto the old transaction's chain. The subject of this new transaction is formed by prepending "Re:" to the old subject. So a transaction chain roughly corresponds to a single conversation.

DISCUSS provides commands that allow you to take advantage of the chaining structure. When DISCUSS prints a transaction, it prints out the "pref" (previous reference) and "nref" (next reference) of that transaction. These are the previous and next transactions within the chain, respectively. By using these numbers, or taking advantage of the "pref" and "nref" transaction specifiers, you can travel through the chain. Other chain-related transaction specifiers are "fref" and "lref", which refer to the beginning and end of the chain.

2.4 Saying something

As stated above, there are two ways you can enter a new transaction in a meeting. To start a new chain, use the "**talk**" command ("**t**" for short), which can also be given as "**enter**" or "**send**".[5] DISCUSS will ask for a subject which will be printed at the head of the message and displayed with "**list**". It will then allow you to type in the transaction. All input up to a "**Control-D**" or a "**.**" on a line by itself will be taken as the text of the message. Once you type in the "**.**" or "**Control-D**", the input will be entered as a new transaction. If you make a mistake during input, don't worry. By typing in "**~e**"[6] on a line by itself, DISCUSS will start up an editor on the text you've entered so

[5] The reasons behind having three different names are mostly historical; Multics Forum had the concept of an "unprocessed" transaction, one for which the text has been entered but has not been sent to a meeting. The "talk" command would start processing (input) of a new transaction, which was not necessarily saved. The "enter" or "send" command would save the unprocessed transaction in the meeting. In DISCUSS, unprocessed transactions have not been implemented.

[6] This is taken from the Berkeley Mail command.

far.[7] Once you have saved the text and exited from the editor, the new transaction will be entered into the meeting.

The other way of entering a transaction is to reply to an existing transaction. For this, use the "**reply**" command ("**rp**" for short). This will ask for input as with "**talk**", except the subject will automatically be taken from the original transaction. The transaction will be entered as a response to the current transaction (by default), or the transaction specified.

2.5 Oops, I didn't mean to say that

If you decide the transaction you just entered is politically incorrect, politically correct, or for some other reason should not be there, it can be deleted. Only the owner of a transaction (that is, the user who posted it), or someone with delete access to the meeting (see the section below on access control), can "**delete**" the transaction. Once deleted, transactions can be retrieved by the owner, or someone with delete access, using the "**retrieve**" command.

2.6 Summary of useful commands

The following is a list of common *discuss* commands, and their abbreviations, as well as a short description of each. This is not necessarily a complete list of available commands.

help — Show help file on a specified command.
list_requests, lr, ? — List available commands.
list_meetings, lsm — List meetings you attend and check for changes.
check_meetings, ckm — Check for changed meetings.
next_meeting, nm — Go to next changed meeting.
add_meeting, add_mtg, am — Add a meeting to your personal list.
delete_meeting, del_mtg, dm — Delete meeting from your personal list.
goto, go, g — Go to the specified meeting.
list, ls — List transactions in a meeting.
print, pr, p — Print a transaction.
next — Print next transaction.
prev — Print previous transation.
nref — Print next reference.
pref — Print previous reference.
read_new, rn — Print new transactions in changed meetings.
talk, enter, send, t — Enter a new transaction.
reply, rp — Reply to a transaction.
write, w — Write a transaction into a file.
delete, dl, d — Delete a transaction.
retrieve, rt — Retrieve a deleted transaction.
status — Show status information.
announce_meeting, ann_mtg, anm — Announce a meeting's existence.
list_acl, la — List access control list.
set_acl, sa — Add principal to access control list.
delete_acl, da — Remove principal from access control list.
quit, exit, q — Quit.

[7] DISCUSS will use the environment variable **EDITOR**, or **/bin/ed** if **EDITOR** is not set. Athena's default startup files set the **EDITOR** environment variable to **emacs**. You may also specify another editor for DISCUSS by setting the environment variable **DISCUSS_EDITOR**.

Discuss provides its own help facility: there is an equivalent of a manual page for each command within Discuss, and also for “access” and “specifiers”. In *discuss*, enter “**help foo**” for information on command or topic “foo”.

3 Using *edsc* — emacs Discuss mode

3.1 Getting started

Discuss can be run within **emacs** by running a sub-process called *edsc*. Like *discuss*, *edsc* can be run over dialup or in a new window. If you are already using **emacs**, enter “**Meta-x discuss**”: hold down on the Meta key (labelled Alt or Compose on many workstations), press the letter “**x**”, then enter the command “**discuss**”. This will start the *edsc* process and load your meetings.

If you want to run Discuss within a new **emacs** editor, enter the command “**emacs -f discuss**” at your **athena%** prompt. The “-f” tells **emacs** to start the *edsc* function automatically.

If this is your first time ever running Discuss, then *edsc* will create a **.meetings** file in which to store information about your meetings. Your personal list of meetings will begin with the two default meetings *New_Meetings* and *Everybody*. The former is for announcements of new meetings, and the latter is for random announcements, questions, and anything else that may be of general interest.

The status line (the reverse-video, penultimate line in the window) should display that the current **emacs** buffer is named ***meetings*** and that Discuss is being run. A message saying “Listing meetings...done.” will be shown in the minibuffer, the final line where messages about what *edsc* is doing are shown and where certain commands are entered. After a short time to load, the main **emacs** window will show something like this:

```
Flags   Meeting name
-----   ------------
c       New_Meetings, new_meetings
c       Everybody, eve
```

The “c” flag means that the meeting has changed so that it contains new unread transactions.

3.2 Adding and reading meetings

The cursor should be next to *New_Meetings*. Aside from being able to move the cursor with the cursor keys or mouse, you can use the keys “**n**” to move to the next changed meeting and “**p**” to move to the previous changed meeting; as normal, “**Ctrl-n**” moves to the next line and “**Ctrl-p**” moves to the previous line.

To attend the *New_Meetings* meeting, simply move the cursor next to *New_Meetings* and type “**g**” to “goto” the meeting. The status line will display that the buffer has changed to the ***New_Meetings meeting*** and will list the number of your current transaction followed by the total number of transactions in the meeting. The main window will show the current transaction.

Similar to when the meetings list was displayed, you can use the keys “**n**” to move to the next transaction and “**p**” to move to the previous transaction. Also, “**Meta-n**” will move to the next transaction in the current chain and “**Meta-p**” will move to the previous transaction in the current chain; chains group transactions by subject. These latter commands allow you to take advantage of the chaining structure of Discuss and more readily follow the topic of conversation. Unlike the *discuss* client, *edsc* is unable to list ranges of transactions.

The typical way to add new meetings is to go to *New_Meetings* and then use the keys "**n**" and "**p**" to view transactions. At a meeting you are interested in adding, press "**a**"; this will automatically add the meeting in the current announcement to your personal list. Pressing "**l**" will then list all your new meetings.

3.3 Saying something

Mentioned above is the idea of chains, sets of transactions having the same subject and forming part of a continuing conversation. When you enter a new transaction into a Discuss meeting, you have the choice of starting a new chain or adding onto an existing chain. To start a new chain while in a meeting, press "**t**" for "talk"; you may then enter the subject and body of your message. To reply to an already existing transaction, press "**r**"; replying will complete the subject field for you by prepending "Re:" to the old subject. After pressing "**r**", the emacs window will split to display the current transaction in the upper half and to allow you to enter your reply in the bottom half.

After you enter the text of your transaction, which can be editted in normal **emacs** fashion, type "**Ctrl-c Ctrl-c**" (yes, twice) to send it to Discuss to be placed in the meeting. If instead you decide not to enter the transaction, typing "**Ctrl-c Ctrl-]**" (right bracket) will cancel your message without putting it in the meeting.

3.4 Summary of useful commands

To get *edsc* to display a help buffer listing available commands, press "**Ctrl-h m**". With the cursor in the help buffer, you can page up and down its contents with "**Meta-v**" and "**Ctrl-v**". With the cursor back in the Discuss buffer, pressing "**Ctrl-x 1**" will hide the help buffer and return the window to the original single Discuss buffer. If you lose the Discuss buffer among other ones, simply entering "**Meta-x discuss**" will return to where you left off. There are separate help buffers for the meetings listing and for inside a meeting, as different commands are available in each. Pressing "**Ctrl-h m**" will automatically provide the appropriate help.

Here is a short summary of the more common commands used at the meetings list:[8]

- **n, spacebar** — Go to the next meeting that has unread transactions.
- **p, delete** — Go to the previous meeting that has unread transactions.
- **Ctrl-n** — Go to the next line in the listing.
- **Ctrl-p** — Go to the previous line in the listing.
- **l** — List all meetings attended.
- **g** — Go to the meeting by the cursor.
- **a** — Add meeting; host and pathname will be requested.
- **d** — Delete meeting; verification will be requested.
- **c** — Catch up; mark all transactions in the meeting as read.
- **q** — Quit Discuss mode.

Here is a short summary of the common commands used from within a meeting:

- **spacebar** — Scroll to next screen of this transaction.

[8] The behavior of the spacebar can be modified so that it automagically "does the right thing": pressing spacebar can enter the next changed meeting, page through to the end of a transaction, and continue to the next transaction; pressing delete would have the opposite effect. To get this behavior, place "**(setq discuss-DWIM t)**" in your **.emacs** file. "DWIM" stands for "Do What I Mean". Alternately, you can press "**Escape Escape spacebar**" (twice, not the Meta key), then at the "Eval:" prompt enter "**(setq discuss-DWIM t)**" to have the same effect. To return to the default behavior, remove the line from your **.emacs** file or press "**Escape Escape spacebar**" and enter "**(setq discuss-DWIM nil)**". This feature is recommended mostly for people who feel comfortable with the more arcane features of **emacs**.

delete — Scroll to previous screen of this transaction.
n — Move to Next transaction in meeting.
p — Move to Previous transaction in meeting.
Meta-n — Move to Next transaction in chain.
Meta-p — Move to Previous transaction in chain.
> — Move to Last transaction in meeting.
< — Move to First transaction in meeting.
g — Goto transaction number specified.
t — Talk; enter a new transaction.
r — Reply to current transaction.
f — Forward this transaction via mail.
c — Catch up. Mark all transactions in this meeting as read.
l — Mark transaction as highest-seen and leave meeting.
d — Delete transaction, and move forward.
Ctrl-d — Delete transaction, and move backward.
R — Retrieve a deleted transaction.
a — Add meeting.
q — Quit meeting.

In addition, pressing "**Ctrl-h k** *key*" will give a description of the function of *key*.

4 Using *xdsc*

The *xdsc* program provides a window-oriented user-friendly interface to the DISCUSS system.[9] Much of the below is derived from the *xdsc* manual page. To see this helpful information, enter "**man xdsc**" at your **athena%** prompt.

4.1 Getting started

To start *xdsc*, you need to be using a workstation running the **X** window system. At a public Athena workstation, enter the command "**xdsc &**" at your **athena%** prompt; the ampersand lets *xdsc* run in the background, freeing your **xterm** to do other work. When the window outline appears, move the window where you want it to be placed and click on a mouse button to set it down.

4.2 Basic screen layout

You will see that *xdsc* displays a single window, divided into five areas:

1. A row of eight buttons, some with pull-down command menus.
2. A text window containing a listing of the meetings which the user attends. This window can be also used to display a listing of the transactions within a single meeting.
3. A single line of text displaying the current status.
4. A row of seven buttons, some with pull-down command menus.

[9] *xdsc* currently uses an *edsc* co-process which does the actual communication with DISCUSS. By default, *xdsc* runs **/usr/athena/etc/edsc**. If you want to use a different **edsc**, set the environment variable **EDSC** to its full pathname.

5. A second text window, used to show the text of the current transaction.

Also, an assortment of popup windows will appear as necessary. These are intended to be self-explanatory.

The three middle lines separating the regions have a small square near their right end. By clicking on and dragging these squares with the mouse, you can adjust the relative sizes of the text panes.

4.3 How to use *xdsc*

This section will provide an introduction on how to use *xdsc*. It explains how to read transactions, how to add and delete meetings from the list of meetings you attend, and how to enter transactions of your own. Each *xdsc* command button is explained.

Upper text pane

The upper text pane initially contains a listing of the meetings you attend. The current meeting, i.e., the one from which you are reading transactions, will have a plus sign ("+") next to it. To change meetings, you can either double-click on a meeting's line with the first mouse button, or use the **Up** and **Down** buttons or the arrow keys as described below. If this is the first time you have used DISCUSS in any form, only two meetings will be listed, namely *New_Meetings* and *Everybody*. The former is for announcements of new meetings, and the latter is for random announcements, questions, and anything else that may be of general interest. You can use the configure button, as described below, to add more meetings.

Changed meetings, those with new unread transactions, will have a "c" next to their name.

This window can also be used, via the **mode** button, to show a list of transactions within the current meeting. In this mode, double-clicking on a transaction's line with the first mouse button will cause the transaction's text to be displayed in the lower window.

Upper control area

The upper control area contains commands of a global nature:

Down Moves the current meeting to the next one with unread transactions. If there is no such meeting, a warning popup will appear.

Up Moves the current meeting to the previous one with unread transactions. If there is no such meeting, a warning popup will appear.

update Queries DISCUSS for an updated meetings list. The new list will reflect any transactions which may have been entered since the last update, as well as any changes you may have made to the list of meetings you attend.

configure Pops up a menu with two entries: **add meeting** and **delete meeting**. These are used to modify the list of meetings you attend. Selecting one of these items will pop up a dialog box with fields to complete. These fields may be cryptic to the novice, but fear not: if the current transaction is the announcement of a new meeting, the fields will already be filled in, and the user need only confirm the action by pressing the **Add** or **Delete** button at the bottom of the dialog box.

The typical way to add new meetings is to enter *New_Meetings* and select the transaction mode (described below). Then use the cursor keys to move through the transactions, pressing

"**Return**" to display a meeting announcement, and then using **add meeting** within the **configure** menu to approve the addition.
You will need to press update before these new meetings will appear in your list of meetings.

mode Pops up a menu with two entries: **transactions** and **meetings**. Selecting either of these makes the upper window display either a list of transactions in the current meeting or a list of meetings attended, as appropriate. Note that while in "transactions" mode, you cannot move between meetings.

show This button is only active while in transactions mode, as it controls which transactions are listed. It pops up a menu with three entries: **unread**, **all**, and **back ten**. **unread** causes the unread transactions to be listed. **all** summons up a list of all transactions in the meeting (Warning: this can take a while!). Finally, **back ten** adds the ten immediately previous transactions to the top of the list, and is usually used for searching backwards for a recent transaction.

HELP Displays a screen of help, briefly explaining what the buttons currently on the screen do. To get rid of this screen, press the **dismiss** button at its bottom.

QUIT Exits the application.

Status line

The status line briefly summarizes what *xdsc* is doing at the moment. It typically lists the current meeting, the range of transaction numbers within this meeting, and the current meeting number. This line is also used for status messages when *xdsc* is doing something which may take a while, such as reading the headers for all the transactions in the *Everybody* meeting.

Lower text pane

The lower text pane contains the text of the current transaction, or is blank if there is no current transaction.

Lower control area

The lower control area contains commands which operate on the current transaction or meeting.

next Moves to the next transaction in the current meeting.

prev Moves to the previous transaction in the current meeting.

Next in chain Moves to the next transaction in the same chain as the current transaction.

Prev in chain Moves to the previous transaction in the same chain as the current transaction.

goto Pops up a three-entry menu used for moving to specific transactions. Selecting **number** pops up a dialog box prompting you for a specific transaction number to view. Selecting **first** or **last** moves to the start or the end of the current meeting, respectively. Selecting **start of chain** or **end of chain** moves to the first or the last transaction in the current chain, respectively.

enter Used for entering a transaction in the current meeting. It pops up a menu with two entries, **reply** and **new transaction**. Selecting **reply** will add the transaction to the chain of the current transaction, while **new** will start a chain on a new subject.

After you select one of these entries, a dialog box will appear with a subject line and a text widget. For replies, the subject line will have a default already filled in, while new transactions will have a blank subject line which the user should fill in. The text widget is a standard Athena text widget, where you can use **emacs** commands to enter the body of your transaction.

When done entering the body of your transaction, press the **Send** button to enter the transaction into the meeting. Press **Abort** if you decide not to send the transaction.

write Used for writing the current transaction to a file. It pops up a dialog box where the user can enter a file name, and pressing the **Write** button causes the transaction to be saved to this file.[10]

4.4 Keyboard equivalents

xdsc has been designed to minimize dependence on a mouse. Nearly every function can be accessed with one or two keystrokes, and the user's hands almost never need to leave the keyboard. The keyboard equivalent for any button is always the first letter of its label, and hitting this key has exactly the same action as pressing the button itself. Note that uppercase and lowercase letters can be distinct.

For example, the lowercase "**n**" and "**p**" keys are synonyms for the **next** and **prev** buttons, for going to the next and previous transactions, while uppercase "**N**" and "**P**" stand for **Next-in-chain** and **Prev-in-chain**, respectively.

If a button triggers a menu, the menu will appear and take focus. Hitting a key corresponding to the first letter of a menu entry will fire off that entry and pop down the menu. Any key which does not match a menu entry will abort the menu and pop it down without any action.

When a simple popup dialog box appears, such as **goto-number**, pressing "**Return**" will make it do its default action. You can abort a dialog box by pressing the "**Escape**" key. For complex dialog boxes, i.e., those with more than one text field, "**Return**" moves focus between the text fields and "**Ctrl-Return**" makes it do its default action.

The arrow keys can be used to move the text caret up and down in the upper text window. Pressing "**Return**" then reads whatever meeting or transaction the caret is sitting on.

Finally, in a way similar to **rn**, the space bar is bound to "do the right thing". If the user is reading a transaction, the space bar will scroll one page down. If at the end of a transaction, it moves to the next transaction, and if at the end of a meeting, it moves to the next changed meeting. If there are no further transactions to read, it does nothing. The "**Delete**" key performs the opposite functions.

5 Finding Interesting Meetings

So you're all set to become addicted to DISCUSS, but you only have two meetings. Here's how to find perhaps dozens of meetings to occupy your time.

5.1 Where to look

Perhaps the best way to find meetings that interest you is to read through all the transactions in the *New_Meetings* meeting. The transactions in *New_Meetings* can be used to add those meetings you wish to attend to your personal list of meetings. Spending time browsing through *New_Meetings*

[10] The greyed out **mail to someone** button is a feature that has not yet been implemented.

can prove to be worth the effort. Another way to learn of interesting meetings is to find out from your friends which ones they attend.

5.2 Some existing meetings

Several popular and long-running meetings are listed below, each with a brief description of the topics typically found in them. The number in brackets corresponds to the transaction number of the meeting anouncement in *New_Meetings* which you can use to add the meeting. Several of these meetings are set up to be the recipients of one or more mailing lists. Such lists are generally set up as read-only; if you wish to reply, then you should send to the mailing list. Reading these archived lists in DISCUSS allows one to decrease the amount of mail received and to save disk space. The two meetings *New_Meetings* and *Everybody* that are created in the default **.meetings** file are ones you will most likely want to keep.

New_Meetings, new_meetings — Announcements of new meetings, and a good place to find the meetings you want to attend.

Everybody, eve — Talk and flamage on topics of general interest.

Central_America, ca [0163] — An offshoot of a previous version called *Panama*. Each primary transaction contains all the **.plan** files that have been changed since the previous day. Some of these start some interesting chains.

Quotes_Misquotes..., quotes [0080] — A collection of funny, embarrassing, witty, or out-of-context quotes made by various people.

Professors_Quote_Board, pqb [0143] — Similar to *quotes*, but reserved for quotes made by professors and other instructors.

mit.bboard [0411] — Recipient of the newsgroup "mit.board", a bulletin board for public announcements, lost & found, for-sale notices, etc.

Athena_MSGS, msgs [0105] — Recipient of the "usermsgs" maillist.

Compact_Disc, cd [0148] — Recipient of the "cd" maillist for discussion and bartering of cd's.

Science Fiction & Fantasy, sff [0256] — Chat about sci-fi and fantasy books and their authors.

Depressing_Thoughts, coatrack [0166] — A "coatrack" on which to leave one's problems, take those worries off one's chest, and sigh.

Cheery_Thoughts, ping [0329]— An alternative to *coatrack* for optimists and others to express the nice things in life. Also look for *Amusing_Thoughts*, *Scary_Thoughts*, *Bogus_Thoughts*, *Flaming_Thoughts*, *Other_Thoughts*, and more.

Discuss_development, discuss [0106] — For discussion (pun intended) of the development of DISCUSS, mention of bugs discovered, changes introduced.

5.3 Your .meetings file

As stated in the introduction, DISCUSS does not keep a central list of all DISCUSS meetings. Instead, it maintains a personal list for each user of the meetings that they attend. This list contains the information needed by the client programs to access the meetings stored on DISCUSS servers, including the host and pathnames to each meeting. To specify a meeting you can use the meeting name, or your own abbreviation. All this information is stored in a file called **.meetings** in your home directory. Without this file, DISCUSS will not be able to locate the meetings you attend.

You can change the name and/or location of this file by setting the environment variable **MEETINGS** to indicate its new pathname.[11] This file is deleted and re-created by Discuss in the process of updating the information in it, so that a system crash or program interruption will not cause a loss of data.[12]

The **.meetings** file contains one line for each meeting you attend, each line being divided into six fields partitioned by colons:

1. Determines what flags are set, "changed" being the most common flag.
2. Time the meeting was last visited or changed.
3. Number of the highest transaction seen.
4. Host on which the meeting is stored.
5. Pathname of the meeting on its host.
6. Names and abbreviations for the meeting, separated by commas.

These fields are re-created each time you use Discuss, and generally you will have no need to edit them directly. However, you may wish to change or add your own abbreviations for meetings' names, which can be done using **emacs** or another editor.

If—when you start a Discuss client or try listing your meetings—you get an error such as "Invalid format in meetings file" or "Can't get list of meetings" or (in *xdsc*) "Segmentation fault", then there is probably an error in your **.meetings** file. Look through it for illegal numbers, misentered paths, and blank lines.

6 Chains

In the Discuss logo pictured on the cover, the letters are linked together by chains. A "chain" is a set of transactions linked together by the same subject.

When a user enters a new transaction into a Discuss meeting, the user has the choice of starting a new chain or adding onto an existing chain. The "talk" command will prompt the user for a subject, and will enter the transaction as the start of a new chain. The "reply" command takes an already existing transaction, and adds the new transaction onto the old transaction's chain. The subject of this new transaction is formed by prepending "Re:" to the old subject. So a transaction chain roughly corresponds to a single conversation within the meeting.

Discuss provides commands that allow you to take advantage of the chaining structure. When Discuss prints a transaction, it displays the pref (previous reference) and nref (next reference) of that transaction. These are the previous and next transactions within the chain, respectively. By using these numbers, or taking advantage of the "pref" and "nref" transaction specifiers, you can travel through the chain and follow the particular conversation. Other chain-related transaction specifiers are "fref" and "lref", which refer to the beginning and end of the chain. Although the various clients have different names for these chaining commands, they all allow for such actions.

7 Searching with *dsgrep*

The other day you saw some transaction about that seminar you wanted to attend, but you forget the room number. Or you can't quite remember what Prof. Mattuck said last week to cause his class to break into laughter. How can you search through the dozens of transactions since then and

[11] To set an environment variable, place a line such as "**setenv MEETINGS /mit/username/meetings_file**" in your **.environment** file; if you do not have a **.environment** file, you can create one with **emacs** or another editor.

[12] Since the meetings file is deleted during each update, however, you should not set it up to refer to a symbolic link, because the link will simply be removed.

find the one you want? To search DISCUSS meetings for a key word or regular expression, you can enter the **dsgrep** command at your **athena%** prompt. This command will print either the title or the contents of matching transactions. For example:

```
athena% dsgrep -e gluco -a -p -i -n 100 pqb
pqb [518]: prof. brown  7.05
"Alpha-D-Glucopyranosyl-(1-4)-Beta-D-Glucopyranose"
"...it is clear, since this is a real compound, that no one would want
to say that..."
*** End of Transaction ***
athena%
```

The options on the command line search the titles and bodies of the last hundred transactions in *Professors_Quote_Board*, ignoring upper/lowercase, for "gluco" and print all matching transactions. The *dsgrep* program has its own manual page describing its use and other options which you can read by entering **man dsgrep** at your **athena%** prompt.

8 Access Control

8.1 Kerberos and DISCUSS

This document is not intended to go into the details of Kerberos; it only explains what you need to do in order to use DISCUSS. Basically you need to be registered with Kerberos, and each time you login you need to be "authenticated" by getting a set of "tickets". The procedure for this is explained below.

- On Athena, you have a Kerberos password if you are able to login to an Athena workstation. You will get Kerberos tickets automatically when you login.
- If you login to a non-Athena timesharing system to use DISCUSS, run **kinit** before using DISCUSS (it will ask for your username and password), and **kdestroy** after you are finished. It's not a bad idea to put **kdestroy -f** in your **.logout** file. If **kinit** fails, contact your Kerberos administrator. At LCS, this is Greg Troxel (username **gdt@lcs.mit.edu**).

If you do not get Kerberos tickets by doing one of the above, you will probably get error messages from DISCUSS mentioning "Kerberos" (such as "Can't read Kerberos ticket file"), and will be treated as an anonymous user by DISCUSS, except for meetings on your local machine. If you have Kerberos tickets you will be treated as a Kerberos-authenticated user.

8.2 Possible access modes

The access modes that can be granted to a user are the following:

A: answer — User can enter replies to existing transactions. Without "own" access, however, he will not be able to delete his own transactions.

C: chairman — Allows the user to modify the access list; this grants the user (indirectly) any access modes he desires.

D: delete — User can delete and retrieve transactions.

O: own — User can read, delete, and retrieve his own submissions to the meeting, i.e., those he entered.

R: read — User can read any (non-deleted) transaction in the meeting.

S: status — User can retrieve certain information about the meeting, such as the number of transactions or access control list.

W: write — User may start new transaction chains in the meeting. Without "own" access, however, he will not be able to delete his transactions.

8.3 Chairman access

The chairman or chairpersons of a meeting can modify the access control list; they therefore can decide who can do what within the meeting. Chairman access does not automatically bypass the other access control terms; thus a chairman can give himself "acorsw" access (add, chairman, own, read, status, and write, but not delete) to prevent himself from accidentally deleting a transaction, although he can explicitly change the access to permit himself to do so.

8.4 Anonymous transactions

Sometimes a transaction will be entered as being from user "???". This means that the authentication (discussed above) failed: the DISCUSS server had no proof of the user's identity. The "???" userid can appear in the access control list, and can be given restricted access; thus, a meeting can be set up to allow an unauthenticated user to read, but require authentication for entering transactions.

9 More Help

Look in sections 2.6 and 3.4 for summaries of useful commands in *discuss* and *edsc*.

9.1 Help files

DISCUSS provides its own help facility: there is an equivalent of a manual page for each command within DISCUSS, and also for "access" and "specifiers". In *discuss*, enter "**help foo**" for information on command or topic "foo". Manual pages are available for *discuss* and *xdsc* by entering "**man discuss**" and "**man xdsc**", respectively. In *edsc*, pressing "**?**" or "**Ctrl-h m**" will give a short summary of commands.

9.2 Who to gripe to...

If you wish to report bugs in the Athena release, please send mail to *bugs@athena.mit.edu*; you may wish to use the "**sendbug**" command at your **athena%** prompt to do this. If you have complaints, suggestions, or any other comments concerning the workings of DISCUSS, please enter them into the *Discuss_development* meeting, or send them via electronic mail to *discussers@athena.mit.edu*. Also feel free to stop by the SIPB office (W20-557, x3-7788) to ask questions.

10 Acknowledgements

Discuss is a project started by Bill Sommerfeld, Ken Raeburn, and Stan Zanarotti of the Student Information Processing Board (SIPB) at MIT. It grew out of an interest in developing a networked version of Forum, the electronic conferencing system on Multics. Multics Forum was originally written by Jay Pattin, J. Spencer Love, and Jeff Schiller, all SIPB members. Stan had also developed a previous conferencing system on VAX/VMS that was named DISCUSS. Discuss borrowed many ideas from these older systems.

Discuss has received generous support from three departments at MIT: the Laboratory for Computer Science, Project Athena, and Information Systems. They allowed the above individuals to spend time on the project, in hopes of a usable system. Other SIPB members have contributed to Discuss, notably Robert French and Ted T'so. Andy Oakland of DCNS deserves credit for writing the *xdsc* client and providing its man page from which parts of this document were written. Lucien W. Van Elsen also deserves credit for writing the *dsgrep* program.

On Athena, over 32,000 transactions (not even including mail feeds) have been entered in more than 200 Discuss meetings. A warm thank you goes out to all users who have made Discuss a successful system.