Portions Copyright (c) 2005 Timothy Daly

The Blue Bayou image Copyright (c) 2004 Jocelyn Guidry

Portions Copyright (c) 2004 Martin Dunstan
Portions Copyright (c) 2007 Alfredo Portes
Portions Copyright (c) 2007 Arthur Ralfs
Portions Copyright (c) 2005 Timothy Daly

Portions Copyright (c) 1991-2002,
The Numerical ALgorithms Group Ltd.
All rights reserved.

This book and the Axiom software is licensed as follows:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of The Numerical ALgorithms Group Ltd. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Inclusion of names in the list of credits is based on historical information and is as accurate as possible. Inclusion of names does not in any way imply an endorsement but represents historical influence on Axiom development.
Contents

1 Overview

2 The axiom Command

2.0.1 [-ht | -noht] ................................................. 3
2.0.2 [-gr | -nogr] .................................................. 3
2.0.3 [-clef | -noclef] ............................................. 4
2.0.4 [-noiw | -iw] .................................................. 4
2.0.5 [-ihere | -noihere] .......................................... 5
2.0.6 [-nox] .......................................................... 5
2.0.7 [-go | -nogo] .................................................. 6
2.0.8 [-ws wsname] .................................................. 6
2.0.9 [-list] .......................................................... 6
2.0.10 [-grprog fname] ............................................. 6
2.0.11 [-htprog fname] ............................................. 6
2.0.12 [-clefprog fname] .......................................... 7
2.0.13 [-sessionprog fname] .................................... 7
2.0.14 [-clientprog fname] ..................................... 7
2.0.15 [-h] .......................................................... 7

3 The sman program

3.1 include files .................................................. 15
3.1.1 include/sman.h ............................................ 16
3.1.2 include/com.h ............................................. 17
3.1.3 include/bsdsignal.h ..................................... 19
3.1.4 include/bsdsignal.h1 ..................................... 19
CONTENTS

3.2.30 print_spad_process_list ........................................... 38
3.2.31 find_child ........................................................... 38
3.2.32 kill_all_children .................................................... 38
3.2.33 clean_up_terminal .................................................... 39
3.2.34 monitor_children ..................................................... 39
3.2.35 main_sman ............................................................ 40
3.2.36 sman.c ............................................................... 41

4 Support Routines ......................................................... 43
   4.1 Command Completion ................................................ 43

5 The viewman program ...................................................... 45

6 The hypertex program ..................................................... 47

7 The clef program .......................................................... 49

8 The session program ...................................................... 51
   8.1 session ............................................................... 51
      8.1.1 includes .......................................................... 51
      8.1.2 variables ........................................................ 51
      8.1.3 usr1_handler ..................................................... 52
      8.1.4 usr2_handler ..................................................... 52
      8.1.5 term_handler .................................................... 53
      8.1.6 pr ................................................................. 53
      8.1.7 close_client ..................................................... 53
      8.1.8 read_SpadServer_command ...................................... 55
      8.1.9 test_sock_for_process .......................................... 55
      8.1.10 read_menu_client_command .................................... 56
      8.1.11 read_from_spad_io .............................................. 57
      8.1.12 kill_spad ........................................................ 57
      8.1.13 accept_session_connection .................................... 58
      8.1.14 read_from_session ............................................. 59
      8.1.15 manage_sessions ............................................... 60
      8.1.16 main_sessionmanager ........................................... 61
CONTENTS

8.1.17 session ......................................................... 62

9 The spadclient program ........................................... 65
  9.1 spadclient ..................................................... 65

10 The Command Completion List ................................. 67

11 Research Topics ................................................ 145
  11.1 Proofs ......................................................... 145
  11.2 Indefinites ................................................... 145
  11.3 Provisos ...................................................... 145

12 Makefile .......................................................... 147

Bibliography .......................................................... 149
New Foreword

On October 1, 2001 Axiom was withdrawn from the market and ended life as a commercial product. On September 3, 2002 Axiom was released under the Modified BSD license, including this document. On August 27, 2003 Axiom was released as free and open source software available for download from the Free Software Foundation's website, Savannah.

Work on Axiom has had the generous support of the Center for Algorithms and Interactive Scientific Computation (CAISS) at City College of New York. Special thanks go to Dr. Gilbert Baumslag for his support of the long term goal.

The online version of this documentation is roughly 1000 pages. In order to make printed versions we’ve broken it up into three volumes. The first volume is tutorial in nature. The second volume is for programmers. The third volume is reference material. We’ve also added a fourth volume for developers. All of these changes represent an experiment in print-on-demand delivery of documentation. Time will tell whether the experiment succeeded.

Axiom has been in existence for over thirty years. It is estimated to contain about three hundred man-years of research and has, as of September 3, 2003, 143 people listed in the credits. All of these people have contributed directly or indirectly to making Axiom available. Axiom is being passed to the next generation. I’m looking forward to future milestones.

With that in mind I’ve introduced the theme of the “30 year horizon”. We must invent the tools that support the Computational Mathematician working 30 years from now. How will research be done when every bit of mathematical knowledge is online and instantly available? What happens when we scale Axiom by a factor of 100, giving us 1.1 million domains? How can we integrate theory with code? How will we integrate theorems and proofs of the mathematics with space-time complexity proofs and running code? What visualization tools are needed? How do we support the conceptual structures and semantics of mathematics in effective ways? How do we support results from the sciences? How do we teach the next generation to be effective Computational Mathematicians?

The “30 year horizon” is much nearer than it appears.

Tim Daly
CAISS, City College of New York
November 10, 2003 ((iHy))
Chapter 1

Overview

The axiom system consists of a set of processes managed by the superman process. The superman process, called sman, is normally invoked from the axiom shell script in order to start a tree of subprocesses.

The axiom command is a shell script that collects the command line options for the sman process, sets some shell variables, and then invokes sman.

The sman process starts the following tree of processes:

```
  --xterm---bash---sman---AXIOMsys
  |---clef---spadclient
  |---hypertex
  |---session
  |---sman
  |---viewman
```
Chapter 2

The axiom Command

The `axiom` command starts everything for Axiom. The options for the `axiom` command are:

```
axiom
[-ht |-noht] whether to use HyperDoc
[-gr |-nogr] whether to use Graphics
[-clef |-noclef] whether to use Clef
[-noiw |-iw] start in interpreter in a separate window
[-ihere |-nohere] start an interpreter in this window
[-nox] don't use X Windows
[-go |-nogo] whether to start system
[-ws wname] use named workspace
[-list] list workspaces only
[-grprog fname] use named program for Graphics
[-htprog fname] use named program for HyperDoc
[-clefprog fname] use named program for Clef
[-sessionprog fname] use named program for session
[-clientprog fname] use named program for spadclient
[-h] show usage
```

In detail, the command options are:

**2.0.1 [-ht | -noht]**

```
[-ht |-noht] whether to use HyperDoc
```

Hyperdoc is the documentation tool for Axiom. The `-ht` option, enabled by default, will start this tool. See Jenks["Jenk92"] Chapter 3 for further information on the hyperdoc subsystem.

**2.0.2 [-gr | -nogr]**

```
[-gr |-nogr] whether to use Graphics
```

The graphics subsystem is enabled using the `-gr` option, enabled by default. Graphics will appear as a result of a draw command, such as

```
draw(sin(x),x=0..1)
```
Note that attempting to use draw commands when the graphics is disabled will simply hang
the interpreter waiting for a response. See Jenks\cite{Jenk92} Chapter 7 for further information
on the graphics subsystem.

\subsection*{2.0.3 \ [-clef | -noclef]}

\[-clef | -noclef\] \hspace{0.5cm} \text{whether to use Clef}

The clef (Command Line Edit Facility) allows for command completion. The list of com-
mand completion strings is in the last chapter of this document. If clef, enabled by default,
is running then you can type:
\begin{verbatim}
x:Dena<tab>
\end{verbatim}
and this will automatically be expanded to:
\begin{verbatim}
x:DenavitHartenbergMatrix
\end{verbatim}

The clef program also allows command line editing. The commands are special keyboard
keys.
\begin{itemize}
  \item HOME move to beginning of the line
  \item END move to the end of the line
  \item CTRL-END delete to end of the line
  \item TAB command completion (multiple tabs give new choices)
  \item UPARROW move back thru commands
  \item DOWNARROW move forward thru commands
  \item LEFTARROW move left on the line
  \item RIGHTARROW move right on the line
  \item INSERT toggle insert/overstrike
\end{itemize}

See Jenks\cite{Jenk92} page 21 for further information on the clef command.

\subsection*{2.0.4 \ [-noiw | -iw]}

\[-noiw | -iw\] \hspace{0.5cm} start in interpreter in a separate window

The iw option, disabled by default, will start a second interpreter in its own window with
its own frame. The fact that the second interpreter is in its own frame can be seen using the
\texttt{)frame} command. For instance, if you type
\begin{verbatim}
axiom -iw
\end{verbatim}
there will be two interpreter windows available, one in the current window and one in a new
window. In the current window if you type:
\begin{verbatim}
)frame names
\end{verbatim}
you will see:
\begin{verbatim}
The names of the existing frames are:
  frame0
  frame1
  initial
The current frame is the first one listed.
\end{verbatim}
In the second window, if you type

)frame names

you will see:

The names of the existing frames are:
frame1
frame0
initial

The current frame is the first one listed.

Setting
x:=3

in the second window will set the variable $x$ in the frame frame1. Switching to the first window and typing:

x

gives:

(1) $x$

Type: Variable $x$

since the first window is in frame0 and the variable $x$ is defined in frame1. But we can switch frames in the first window using

)frame next

and then

x

gives:

(2) 3

Type: PositiveInteger

and now the two windows share the same frame space. See Jenks\cite{Jenk92} page 579 for further information on the \texttt{frame} command.

2.0.5 \([-ihere | -noihere]\)

\[-ihere | -noihere\] start an interpreter in this window

This option determines whether Axiom will start in the current window. Using this option alone is not particularly useful and it is generally used in combination with the \texttt{-iw} option:

axiom -noihere -iw &

However, used alone, as in:

axiom -noihere &

it will start Axiom and show the Hyperdoc window. Graphics will also work from the Hyperdoc pages.

2.0.6 \([-nox]\)

\[-nox\] don't use X Windows
allows Axiom to start the interpreter without Hyperdoc or the graphics subsystem. This is useful for starting Axiom in an emacs buffer.

### 2.0.7 [\[-go \mid -nogo\]

\[-go \mid -nogo\] whether to start system

uses the \[-go\] option, enabled by default, controls whether the system starts from the command line. If the \[-nogo\] option is chosen the system prints the command line that would have been issued. This is useful for finding out what the command line options to \sman\ will be. For instance:

```
axiom -nogo -iw
```
does not start Axiom but types out:

Would now start the processes.
exec `/mnt/linux/bin/sman -iw -ws `/mnt/linux/bin/AXIOMsys`

### 2.0.8 [\[-ws wsname\]

\[-ws wsname\] use named workspace

In the \[-nogo\] command above you can see that the default workspace name is

\[-ws `/mnt/linux/bin/AXIOMsys`

This option allows you to change that. This is useful for debugging new system builds. During build a debugging version of Axiom is created in the \obj/linux/bin\ directory. The \debugsys\ image uses interpreted lisp code rather than compiled code. This makes it possible to do deep debugging. To use this workspace you would incant:

```
cd youraxiombuild
eexport AXIOM='pwd'/mnt/linux
export PATH=$AXIOM/bin:$PATH
axiom -ws obj/linux/bin/debugsys
```

### 2.0.9 [\[-list\]

\[-list\] list workspaces only

shows you the executable workspaces. Generally in a built system there is only one, called \$AXIOM/bin/AXIOMsys\.

### 2.0.10 [\[-grprog fname\]

\[-grprog fname\] use named program for Graphics

allows you to specify which program to use for the graphics. By default this is \$AXIOM/lib/viewman\.

### 2.0.11 [\[-htprog fname\]

\[-htprog fname\] use named program for Hyperdoc
allows you to specify which program to use for Hyperdoc. By default it is
$AXIOM/bin/hypertex -s.

2.0.12 [-clefprog fname]

[-clefprog fname] use named program for Clef
allows you to specify which program to use for clef. By default it is
$AXIOM/bin/clef -f $AXIOM/lib/command.list -e.

2.0.13 [-sessionprog fname]

[-sessionprog fname] use named program for session
allows you to specify the session manager program. By default it is
$AXIOM/lib/session.

2.0.14 [-clientprog fname]

[-clientprog fname] use named program for spadclient
allows you to specify the spadclient program. By default it is
$AXIOM/lib/spadclient.

2.0.15 [-h]

[-h] show usage

— axiomcmd —

#!/bin/sh

The MALLOCTYPE shell variable is an IBM AIX shell variable that controls buckets based
extensions in the default memory allocator which may enhance performance. AIX uses a
new memory management routine that does not zero malloc memory and does not round
up to the nearest power of 2, unlike most non-AIX systems. This can cause failures so we
protect against that here. See the AIX Performance Tuning Guide[Haya05] for details.

— axiomcmd —

MALLOCTYPE=3.1
export MALLOCTYPE

— axiomcmd —

HOST='hostname'
export HOST
CHAPTER 2. THE AXIOM COMMAND

There are 4 basic utilities used by this script. The ciao script for immediate exit:

```bash
-- axiomcmd --
ciao() {
  echo "Goodbye."
  exit 1
}
```

The needsubopt script which is used to issue an error message when one of the command line options requires an option:

```bash
-- axiomcmd --
needsubopt () {
  echo "The $1 option requires an argument."
  ciao
}
```

The showuse script which gives basic command line help:

```bash
-- axiomcmd --
showuse() {
  echo "axiom"
  echo "  [-ht | -noht] whether to use HyperDoc"
  echo "  [-gr | -nogr] whether to use Graphics"
  echo "  [-clef | -noclef] whether to use Clef"
  echo "  [-noiw | -iw] start in interpreter in a separate window"
  echo "  [-ihere | -noihere] start an interpreter in this window"
  echo "  [-nox] don't use X Windows"
  echo "  [-go | -nogo] whether to start system"
  echo "  [-ws wsname] use named workspace"
  echo "  [-list] list workspaces only"
  echo "  [-grprog fname] use named program for Graphics"
  echo "  [-htprog fname] use named program for HyperDoc"
  echo "  [-clefprog fname] use named program for Clef"
  echo "  [-sessionprog fname] use named program for session"
  echo "  [-clientprog fname] use named program for spadclient"
  echo "  [-h] show usage"
}
```

List the various workspaces if asked.

```bash
-- axiomcmd --
listwspaces()
{
  echo "$1"
  ls -l $2 | grep "sys$"
  echo ""
}
```
Step 1. Ensure the environment is set.

Just process “-h”. If it exists in the command line then we print out the simple command line help menu.

```bash
-- axiomcmd --
if [ "$*" = "-h" ] ; then
  showuse
fi
```

We assume that Axiom is installed in the standard place on a Linux system. We will modify this assumption as we process the environment and command line. The term `spad` is an historical shortened version of the name `scratchpad`, the original name of the Axiom system.

```bash
-- axiomcmd --
SPADDEFAULT=/usr/local/axiom/mnt/linux
```

If the `$AXIOM` shell variable is set then we use it.
If not, then if the `$SPAD` shell variable is set then we use it.
If not, then we try to use the default value above.
If not, we simply fail.

```bash
-- axiomcmd --
if [ "$SPAD" = "" ] ; then
  if [ "$AXIOM" = "" ] ; then
    SPAD=$SPADDEFAULT
    echo "AXIOM variable is not set"
    echo "assuming AXIOM = $SPAD"
    AXIOM=$SPAD
    export AXIOM
  else
    SPAD=$AXIOM
  fi
  export SPAD
else
  if [ "$AXIOM" = "" ] ; then
    echo "AXIOM variable is not set"
    echo "but SPAD = $SPAD"
    echo "Using AXIOM = $SPAD"
    AXIOM=$SPAD
    export AXIOM
  else
    if [ ! "$SPAD" = "$AXIOM" ] ; then
      echo "ignoring SPAD variable"
      SPAD=$AXIOM
    fi
  fi
fi
```
If we get here then all attempts to find axiom have failed so we complain and exit.

```bash
if [ ! -d "$SPAD" ] ; then
echo "The directory for Axiom, $SPAD, does not exist."
ciao
fi
```

---

Step 2. Process command line arguments.

Name the workspace directories

```bash
rootwsdir=$SPAD/bin
```

---

We set up the defaults for command-line arguments. We don’t want just a list by default

```bash
list=no
```

---

We default to actually executing the workspace.

```bash
go=yes
```

---

We default to the `AXIOMsys` workspace.

```bash
wsname=AXIOMsys
```

---

And all other options are unset.

```bash
otheropts=""
```

---

For each option on the command line do

```bash
while [ "$*" != "" ] ; do
    case $1 in
```
If the user specified list anywhere then we give the workspace list and exit.

```bash
--- axiomcmd ---
  -list) list=yes
      go=no;;
```  
If the user specified go or nogo we handle that case

```bash
--- axiomcmd ---
  -go) go=yes ;;
  -nogo) go=no ;;
```  
The workspace option requires an argument which follows immediately. If the argument is missing we complain and exit.

```bash
--- axiomcmd ---
  -ws) if [ "$2" = "" ] ; then needsubopt "$1" ; fi
  shift
  wname="$1"
  ;;
```  
We can specify the various subprograms to use.

```bash
--- axiomcmd ---
  -grprog|-htprog|-clefprog|-sessionprog|-clientprog)
  if [ "$2" = "" ] ; then needsubopt "$1" ; fi
  otheropts="$otheropts $1 $2"
  shift
  ;;
```  
These options were not explained earlier and are only for developer use.

```bash
--- axiomcmd ---
  -paste|-rm|-rv)
  if [ "$2" = "" ] ; then needsubopt "$1" ; fi
  otheropts="$otheropts $1 $2"
  shift
  ;;
```  
We handle the various [-option | -nooption] cases

```bash
--- axiomcmd ---
  -clef|-noclef|-gr|-n ogr|-ht|-noht|-iw|-noiw)
  otheropts="$otheropts $1"
  ;;
  -ihere|-noihere|-nox)
  otheropts="$otheropts $1"
  ;;
```
The user wanted help so we will not execute.

```bash
  axiomcmd
  -h)
go=no
;;
```

The user is confused. Complain and exit.

```bash
  axiomcmd
  *) echo "Unknown option: $1"
echo "To use a specific workspace use, e.g.: spad -ws $1"
ciao
;;
esac
```

Move to the next option and loop.

```bash
  axiomcmd
  shift
done
```

Step 3. Handle options that require special case handling.

The user just wanted to know what workspaces are available.

```bash
  axiomcmd
  if [ $list = yes ] ; then
    listwspaces "AXIOM workspaces in \$AXIOM/bin = $rootwsdir: " $rootwsdir
  fi
```

Try to ensure a suitable workspace on this host.

```bash
  axiomcmd
  if [ `expr $wsname : '.*/.*'` = 0 ] ; then
    serverws=$rootwsdir/$wsname
  else
    serverws=$wsname
  fi
```

If we can’t find the executable then we complain and exit.

```bash
  axiomcmd
  if [ ! -x $serverws ] ; then
    echo "Cannot find the executable $serverws"
    showuse
    ciao
    fi
```
The user just wanted to see what would happen so we output the command line and exit.

```bash
| axiomcmd |

if [ $go = no ] ; then
    echo "Would now start the processes."
    echo exec $SPAD/bin/sman $otheropts -ws $serverws
    exit 0
fi
```

All of the options have been processed so we start `sman`.

```bash
| axiomcmd |

exec $SPAD/bin/sman $otheropts -ws $serverws
```
Chapter 3

The sman program

3.1 include files

.include/sockio-c.h1

/* from bookvol6 chunk include/sockio-c.h1 */
extern int get_int(Sock *);
extern char * get_string(Sock *);
extern double get_float(Sock *);
extern Sock * connect_to_local_server(char * , int , int);
extern int sread(Sock * , char * , int , char *);
extern double plus_infinity(void);
extern double minus_infinity(void);
extern double NANQ(void);
extern void sigpipe_handler(int);
extern int wait_for_client_read(Sock * , char * , int , char *);
extern int wait_for_client_write(Sock * , char * , int , char *);
extern int swrite(Sock * , char * , int , char *);
extern int sselect(int , fd_set * , fd_set * , fd_set * , void *);
extern int fill_buf(Sock * , char * , int , char *);
extern int sock_get_int(int);
extern int get_ints(Sock * , int * , int);
extern int sock_get_ints(int , int * , int);
extern int send_int(Sock * , int);
extern int sock_send_int(int , int);
extern int send_ints(Sock * , int * , int);
extern int sock_send_ints(int , int * , int);
extern int send_string(Sock * , char *);
extern int send_string_len(Sock * , char * , int);
extern int sock_send_string(int , char *);
extern int sock_send_string_len(int , char * , int);
extern int send_strings(Sock * , char ** , int);
extern int sock_send_strings(int , char ** , int);
extern char * sock_get_string(int);
extern char * get_string_buf(Sock * , char * , int);
extern char * sock_get_string_buf(int , char * , int);
extern int get_strings(Sock * , char * * , int );
extern int sock_get_strings(int , char * * , int );
extern int send_float(Sock * , double );
extern int sock_send_float(int , double );
extern int send_sfloats(Sock * , float * , int );
extern int sock_send_sfloats(int , float * , int );
extern int send_floats(Sock * , double * , int );
extern int sock_send_floats(int , double * , int );
extern double sock_get_float(int );
extern int get_sfloats(Sock * , float * , int );
extern int sock_get_sfloats(int , float * , int );
extern int get_floats(Sock * , double * , int );
extern int sock_get_floats(int , double * , int );
extern int wait_for_client_kill(Sock * , int );
extern int sock_get_remote_fd(int );
extern int send_signal(Sock * , int );
extern int sock_send_signal(int , int );
extern int send_wakeup(Sock * );
extern int sock_send_wakeup(int );
extern Sock * connect_to_local_server_new(char * , int , int );
extern void remote_stdio(Sock * );
extern void init_purpose_table(void );
extern int make_server_number(void );
extern void close_socket(int , char * );
extern int make_server_name(char * , char * );
extern int open_server(char * );
extern int accept_connection(Sock * );
extern void get_socket_type(Sock * );
extern int sock_accept_connection(int );
extern void redirect_stdio(Sock * );
extern void init_socks(void );
extern int server_switch(void );
extern void flush_stdout(void );
extern void print_line(char * );

3.1.1 include/sman.h

The spad_proc structure holds information about the process id of a child process, what to do when it dies, and the shell command line necessary to restart the process. There is a linked list of these structures which maintains the process list for axiom.

#include/sman.h —

/* from bookvol6 chunk include/sman.h */
/* Process control definitions. Used by fork_you and spawn_of_hell */

/* When a process dies it kills off everything else */
#define Die 1
/* When a process dies, do nothing */
#define NadaDelShitsky 2
/* When a process dies start it up again */
#define DoItAgain 3
/* When hypertex dies, clean its socket */
3.1. INCLUDE FILES

#define CleanHypertexSocket 4

typedef struct spad_proc {
   int proc_id; /* process id of child */
   int death_action; /* one of the above constants */
   char *command; /* sh command line to restart the process */
   struct spad_proc *next;
} SpadProcess;

3.1.2 include/com.h

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#if defined(RIOSplatform)
#include <sys/select.h>
#endif

typedef struct {
   int socket; /* socket number returned by "socket" call */
   int type; /* socket type (AF_UNIX or AF_INET) */
   int purpose; /* can be SessionManager, GraphicsServer, etc. */
   int pid; /* process ID of connected socket */
   int frame; /* spad interpreter frame (for interpreter windows) */
   int remote_fd; /* file descriptor of remote socket */
   union {
      struct sockaddr u_addr;
      struct sockaddr_in i_addr;
   } addr;
   char *host_name; /* name of foreign host if type == AF_INET */
} Sock;

#define MaxClients 150

/* possible socket types (purpose) */

#define SessionManager 1
#define ViewportServer 2
#define MenuServer 3
#define SessionIO 4
#define BalloonServer 5
#define InterpWindow 6
#define KillSpad 7
#define DebugWindow 8
#define Forker 9
#define AV 10 /* Simon's algebraic viewer */

#define Acknowledge 255
/* Timeout value for connection to remote socket */
#define Forever 0
/* Socket name for local AXIOM server and session manager */

#define SpadServer "/tmp/.d"
#define SessionServer "/tmp/.s"
#define SessionIOName "/tmp/.i"
#define MenuServerName "/tmp/.h"
#define ForkServerName "/tmp/.f"

#define MASK_SIZE (NBBY*sizeof(fd_set))

/* table of dedicated socket types */
extern Sock *purpose_table[];
extern Sock server[];
extern Sock clients[];
extern fd_set socket_mask;
extern fd_set server_mask;

/* Commands sent over the AXIOM session manager or menu socket */
#define CreateFrame 1
#define SwitchFrames 2
#define EndOfOutput 3
#define CallInterp 4
#define EndSession 5
#define LispCommand 6
#define SpadCommand 7
#define SendXEventToHyperTeX 8
#define QuietSpadCommand 9
#define CloseClient 10
#define QueryClients 11
#define QuerySpad 12
#define NonSmanSession 13
#define KillLispSystem 14

#define CreateFrameAnswer 50
/* Commands from AXIOM menu server to interpreter windows */
#define ReceiveInputLine 100
#define TestLine 101

#endif
3.1.3 include/bsdsignal.h

---

3.1.3.1 include/bsdsignal.h

/* from bookvol6 chunk include/bsdsignal.h */
#ifndef _BSDSIGNAL_H_
#define _BSDSIGNAL_H_

#define RestartSystemCalls 1
#define DontRestartSystemCalls 0

typedef void (* SignalHandlerFunc)(int);
#endif /* _BSDSIGNAL */
---

3.1.4 include/bsdsignal.h1

---

3.1.4.1 include/bsdsignal.h1

/* from bookvol6 chunk include/bsdsignal.h1 */
extern SignalHandlerFunc bsdSignal(int , SignalHandlerFunc , int );
---

3.1.5 include/openpty.h

---

3.1.5.1 include/openpty.h

/* from bookvol6 chunk include/openpty.h */
extern void makeNextPtyNames(char * , char * );
extern int ptyopen(int * , int * , char * , char * );
---

3.1.6 include/sman.h

---

3.1.6.1 include/sman.h

/* from bookvol6 chunk include/sman.h */
extern int main(int argc , char * argv[], char * envp[]);
#endif _SMAN_C
static void process_arguments(int argc , char * * argv);
static int should_I_clef(void);
static int in_X(void);
static void set_up_defaults(void);
static void process_options(int argc , char * * argv);
static void death_handler(int sig);
static void sman_catch_signals(void);
static void fix_env(char * * envp , int spadnum);
static void init_term_io(void);
static char * strPrefix(char * prefix , char * s);
static void check_spad_proc(char * file , char * prefix);
static void clean_up_old_sockets(void);
static SpadProcess * fork_you(int death_action);
static void exec_command_env(char * command , char * * env);
static SpadProcess * spawn_of_hell(char * command , int death_action);
static void start_the_spadclient(void);
static void start_the_local_spadclient(void);
static void start_the_session_manager(void);
static void start_the_hypertex(void);
static void start_the_graphics(void);
static void fork_Axiom(void);
static void start_the_Axiom(char * * envp);
static void clean_up_sockets(void);
static void clean_hypertex_socket(void);
static void read_from_spad_io(int ptcNum);
static void read_from_manager(int ptcNum);
static void manage_spad_io(int ptcNum);
static void init_spad_process_list(void);
static SpadProcess * find_child(int proc_id);
static void kill_all_children(void);
static void clean_up_terminal(void);
static void monitor_children(void);
#endif

3.1.7 include/session.h1

#include/session.h1
/* from bookvol6 chunk include/session.h1 */
extern int main(void);
#ifdef _SESSION_C
static void usr1_handler(int sig);
static void usr2_handler(int sig);
static void term_handler(int sig);
static void close_client(int frame);
static void read_SpadServer_command(void);
static int test_sock_for_process(Sock * sock);
static void read_menu_client_command(void);
static void read_from_spad_io(void);
static void kill_spad(void);
static int accept_session_connection(Sock * server_sock);
static void read_from_session(Sock * sock);
static void manage_sessions(void);
#endif

3.2 sman.c

3.2.1 includes

---

---

3.2.2 variables

---
We add a debug flag so we can print information about what sman is trying to do. This change is pervasive as it touches nearly every routine.

— sman.variables —

```c
int tpd=0;    /* to-print-debug information */

/**********************************************************
/* definitions of programs which sman can start */
/**********************************************************/

#define SpadDefaultFile "spadprof.input"

char ClefCommandLine[256];

#define BufSize  4096    /* size of communication buffer */
char big_bad_buf[BufSize];     /* big I/O buffer */

Sock *session_io = NULL;      /* socket connecting to session manager */

/**********************************************************
/* Some characters used and externally defined in edible.h */
/**********************************************************/

unsigned char _INTR, _QUIT, _ERASE, _KILL, _EOF, _EOL, _RES1, _RES2;

/**********************************************************
/* Stuff for opening pseudo-terminal */
/**********************************************************/

int ptsNum, ptcNum;
char ptsPath[20], ptcPath[20];

typedef char **new_envp;    /* new environment for AXIOM */
int child_pid;       /* child’s process id */
```
struct termios oldbuf;    /* the original settings */
struct termios childbuf;  /* terminal structure for user i/o */

int death_signal = 0;


3.2.3  process_arguments

— sman.processarguments —

static void
process_arguments(int argc,char ** argv)
{
  int arg;
  if (tpd == 1) fprintf(stderr,"sman:process_arguments entered\n");
  for (arg = 1; arg < argc; arg++)
    if (strcmp(argv[arg], "-debug") == 0)
      tpd = 1;
    else if (strcmp(argv[arg], "-noclef") == 0)
      start_clef = 0;
    else if (strcmp(argv[arg], "-clef") == 0)
      start_clef = 1;
    else if (strcmp(argv[arg], "-gr") == 0)
      start_graphics = 1;
    else if (strcmp(argv[arg], "-nogr") == 0)
      start_graphics = 0;
    else if (strcmp(argv[arg], "-ht") == 0)
      start_ht = 1;
    else if (strcmp(argv[arg], "-noht") == 0)
      start_ht = 0;
    else if (strcmp(argv[arg], "-iw") == 0)
      start_spadclient = 1;
    else if (strcmp(argv[arg], "-ihere") == 0)
      start_local_spadclient = 1;
    else if (strcmp(argv[arg], "-noiw") == 0)
      start_spadclient = 0;
    else if (strcmp(argv[arg], "-noihere") == 0)
      start_local_spadclient = 0;
    else if (strcmp(argv[arg], "-ws") == 0)
      use_X = 0;
    else if (strcmp(argv[arg], "-comp") == 0)
      ws_path = "$AXIOM/etc/images/comp";
    else if (strcmp(argv[arg], "-nocomp") == 0)
      ws_path = "$AXIOM/etc/images/comp";
  }
  else if (strcmp(argv[arg], "-grprog") == 0)
GraphicsProgram = argv[++arg];
else if (strcmp(argv[arg], "-htprog") == 0)
    HypertextProgram = argv[++arg];
else if (strcmp(argv[arg], "-clefprog") == 0) {
    strcpy(ClefCommandLine, argv[++arg]);
    ClefProgram = strcat(ClefCommandLine, " -f $AXIOM/lib/command.list -e ");
} else if (strcmp(argv[arg], "-sessionprog") == 0)
    SessionManagerProgram = argv[++arg];
else if (strcmp(argv[arg], "-clientprog") == 0)
    SpadClientProgram = argv[++arg];
else if (strcmp(argv[arg], "-rm") == 0)
    MakeRecordFile = argv[++arg];
else if (strcmp(argv[arg], "-rv") == 0)
    VerifyRecordFile = argv[++arg];
else if (strcmp(argv[arg], "-paste") == 0)
    PasteFile = argv[++arg];
else {
    fprintf(stderr, "Usage: sman <-clef|-noclef> <-gr|-nogr> <-ht|-noht>");
    fprintf(stderr, " <-iw|-noiw> <-nox> <-comp>");
    fprintf(stderr, " <-ws spad_workspace> <-grprog path> <-htprog path>");
    fprintf(stderr, " <-clefprog path> <-sessionprog path>");
    exit(-1);
}
if (tpd == 1) {
    fprintf(stderr," sman ");
    if (start_clef == 0)
        fprintf(stderr,"-noclef ");
    else
        fprintf(stderr,"-clef ");
    if (start_graphics == 0)
        fprintf(stderr,"-nogr ");
    else
        fprintf(stderr,"-gr ");
    if (start_ht == 0)
        fprintf(stderr,"-noht ");
    else
        fprintf(stderr,"-ht ");
    if (start_spadclient == 0)
        fprintf(stderr,"-noiw ");
    else
        fprintf(stderr,"-iw ");
    if (start_local_spadclient == 0)
        fprintf(stderr,"-noihere ");
    else
        fprintf(stderr,"-ihere ");
    if (use_X == 0)
3.2. SMAN.C

    fprintf(stderr,"-nox ");
    fprintf(stderr,"-ws ");
    fprintf(stderr,"%s ",ws_path);
    fprintf(stderr,"-grprog ");
    fprintf(stderr,"%s ",GraphicsProgram);
    fprintf(stderr,"-htprog ");
    fprintf(stderr,"%s ",HypertextProgram);
    fprintf(stderr,"-clefprog ");
    fprintf(stderr,"%s ",ClefCommandLine);
    fprintf(stderr,"-sessionprog ");
    fprintf(stderr,"%s ",SessionManagerProgram);
    fprintf(stderr,"-clientprog ");
    fprintf(stderr,"%s ",SpadClientProgram);
    fprintf(stderr,"-rm ");
    fprintf(stderr,"%s ",MakeRecordFile);
    fprintf(stderr,"-rv ");
    fprintf(stderr,"%s ",VerifyRecordFile);
    fprintf(stderr,"-paste ");
    fprintf(stderr,"%s ",PasteFile);
    fprintf(stderr,"\n");
    if (tpd == 1) fprintf(stderr,"sman:process_arguments exit\n");
}

3.2.4 should_I_clef

    sman.shouldIclef —

static int
should_I_clef(void)
{
    return(1);
}

3.2.5 in_X

    sman.inX —

static int
in_X(void)
{
    if (getenv("DISPLAY")) return 1;
    return 0;
}
3.2.6 set_up_defaults

These are the default values for sman. A '1' value means that sman will try to start the given process, a '0' value means not starting the process.

--- sman.setupdefaults ---

```c
static void
set_up_defaults(void)
{
    if (tpd == 1) fprintf(stderr,"sman:setupdefaults entered\n");
    start_clef = should_I_clef();
    start_graphics = 1;
    start_ht = 1;
    start_spadclient = 0;
    start_local_spadclient = 1;
    use_X = isatty(0) && in_X();
    ws_path = "$AXIOM/bin/AXIOMsys";
    if (tpd == 1) fprintf(stderr,"sman:setupdefaults exit\n");
}
---
```

3.2.7 process_options

--- sman.processoptions ---

```c
static void
process_options(int argc, char **argv)
{
    if (tpd == 1) fprintf(stderr,"sman:process_options entered\n");
    set_up_defaults();
    process_arguments(argc, argv);
    if (tpd == 1) fprintf(stderr,"sman:process_options exit\n");
}
---
```

3.2.8 death_handler

--- sman.deathhandler ---

```c
static void
death_handler(int sig)
{
    death_signal = 1;
}
---
```
3.2.9 sman_catch_signals

— sman.smancatchsignals —

static void
sman_catch_signals(void)
{
    /* Set up the signal handlers for sman */
    bsdSignal(SIGINT, SIG_IGN, RestartSystemCalls);
    bsdSignal(SIGTERM, death_handler, RestartSystemCalls);
    bsdSignal(SIGQUIT, death_handler, RestartSystemCalls);
    bsdSignal(SIGHUP, death_handler, RestartSystemCalls);
    bsdSignal(SIGILL, death_handler, RestartSystemCalls);
    bsdSignal(SIGTRAP, death_handler, RestartSystemCalls);
    bsdSignal(SIGIOT, death_handler, RestartSystemCalls);
    bsdSignal(SIGBUS, death_handler, RestartSystemCalls);
    bsdSignal(SIGSEGV, death_handler, RestartSystemCalls);
    bsdSignal(SIGUSR1, death_handler, RestartSystemCalls);
}

3.2.10 fix_env

insert SPADSERVER and SPADNUM variables into the environment
— sman.fixenv —

static void
fix_env(char **envp, int spadnum)
{
    int len, i;
    char *sn;
    for(len = 0; envp[len] != NULL; len++);
    new_envp = (char **) malloc((len + 3) * sizeof(char *));
    new_envp[0] = "SPADSERVER=TRUE";
    sn = (char *) malloc(20 * sizeof(char));
    sprintf(sn, "SPADNUM=%d", spadnum);
    new_envp[1] = sn;
    for(i=0; i<len; i++)
        new_envp[i+2] = envp[i];
}

3.2.11 init_term_io

— sman.inittermio —

static void
init_term_io(void)
{
    if(!isatty(0)) return;
    if( tcgetattr(0, &oldbuf) == -1) {
        perror("getting termios");
        return; /* exit(-1); */
    }
    if( tcgetattr(0, &childbuf) == -1) {
        perror("getting termios");
        return; /* exit(-1); */
    }
    _INTR = oldbuf.c_cc[VINTR];
    _QUIT = oldbuf.c_cc[VQUIT];
    _ERASE = oldbuf.c_cc[VERASE];
    _KILL = oldbuf.c_cc[VKILL];
    _EOF = oldbuf.c_cc[VEOF];
    _EOL = oldbuf.c_cc[VEOL];
}

3.2.12  strPrefix

--- sman.strPrefix ---

static char *
strPrefix(char *prefix,char * s)
{
    while (*prefix != '\0' && *prefix == *s) {
        prefix++;
        s++;
    }
    if (*prefix == '\0') return s;
    return NULL;
}

---

3.2.13  check_spad_proc

--- sman.checkspadproc ---

static void
check_spad_proc(char *file, char *prefix)
{
    char *num;
    int pid;
    if (!(*num = strPrefix(prefix, file))) {
        pid = atoi(num);
        if (pid > 2) {
            
```---
```
3.2. SMAN.C

```c
    kill(pid, 0);
    if (kill(pid, 0) == -1 && errno == ESRCH) {
        unlink(file);
    }
}
```
CHAPTER 3. THE SMAN PROGRAM

{ /* fork a new process, giving it a default death action */
 /* return NULL in child, SpadProcess in parent */
 int child_pid = fork();
 SpadProcess *proc;
 if (child_pid) return NULL;
 proc = (SpadProcess *) malloc(sizeof(SpadProcess));
 proc->proc_id = child_pid;
 proc->death_action = death_action;
 proc->command = NULL;
 proc->next = spad_process_list;
 spad_process_list = proc;
 return proc;
}

3.2.16 exec_command_env

Note that the next-to-last argument of execle must be an explicit NULL pointer. The previous naked 0 value was not correct.

— sman.execcommandenv —

static void
exec_command_env(char *command,char ** env)
{
    char new_command[512];
    sprintf(new_command, "exec \%s", command);
    execle("/bin/sh","/bin/sh", "-c", new_command, (char *)0, env);
}

3.2.17 spawn_of_hell

— sman.spawnofhell —

static SpadProcess *
spawn_of_hell(char *command, int death_action)
{
    SpadProcess *proc = fork_you(death_action);
    if (proc != NULL) {
        proc->command = command;
        return proc;
    }
    exec_command_env(command, new_envp);
    return NULL;
}
3.2.18 start_the_spadclient

run a AXIOM client in the main process

--- sman.startthespadclient ---

static void
start_the_spadclient(void)
{
  char command[256];
  if (start_clef)
    #ifdef RIOSplatform
      sprintf(command,
      "aixterm -sb -sl 500 -name axiomclient -n AXIOM -T AXIOM -e \%s %s",
      ClefProgram, SpadClientProgram);
    #else
      sprintf(command,
      "xterm -sb -sl 500 -name axiomclient -n AXIOM -T AXIOM -e %s %s",
      ClefProgram, SpadClientProgram);
    #endif
    else
      #ifdef RIOSplatform
        sprintf(command,
        "aixterm -sb -sl 500 -name axiomclient -n AXIOM -T AXIOM -e %s",
        SpadClientProgram);
      #else
        sprintf(command,
        "xterm -sb -sl 500 -name axiomclient -n AXIOM -T AXIOM -e %s",
        SpadClientProgram);
      #endif
      if (tpd == 1)
        fprintf(stderr,"sman:start_the_spadclient: %s\n",command);
    spawn_of_hell(command, NadaDelShitsky);
}

3.2.19 start_the_local_spadclient

--- sman.startthelocalspadclient ---

static void
start_the_local_spadclient(void)
{
  char command[256];
  if (start_clef)
    sprintf(command, "%s %s", ClefProgram, SpadClientProgram);
  else
    sprintf(command, "%s", SpadClientProgram);
  if (tpd == 1)
    fprintf(stderr,"sman:start_the_local_spadclient: %s\n",command);
  spawn_of_hell(command, NadaDelShitsky);
}
### 3.2.20 start_the_session_manager

---

**— sman.startthesessionmanager —**

```c
static void
start_the_session_manager(void)
{
    spawn_of_hell(SessionManagerProgram, Die);
}
```

---

### 3.2.21 start_the_hypertex

---

**— sman.startthehypertex —**

```c
static void
start_the_hypertex(void)
{
    char prog[512];

    if (PasteFile){
        sprintf(prog, "%s -k -ip %s", HypertexProgram, PasteFile);
        spawn_of_hell(prog, NadaDelShitsky);
    }
    else if (MakeRecordFile){
        sprintf(prog, "%s -k -rm %s", HypertexProgram, MakeRecordFile);
        spawn_of_hell(prog, NadaDelShitsky);
    }
    else if (VerifyRecordFile){
        sprintf(prog, "%s -k -rv %s", HypertexProgram, VerifyRecordFile);
        spawn_of_hell(prog, NadaDelShitsky);
    }
    /* If we restart hyperdoc from the axiom command prompt */
    else spawn_of_hell(HypertexProgram, CleanHypertexSocket);
}
```

---

### 3.2.22 start_the_graphics

---

**— sman.startthegraphics —**

```c
static void
start_the_graphics(void)
```
3.2. SMAN.C

{
    spawn_of_hell(GraphicsProgram, DoItAgain);
}

3.2.23 fork_Axiom

— sman.forkAxiom —

/* Start the AXIOM session in a separate process, */
/* using a pseudo-terminal to catch all input and output */
static void
fork_Axiom(void)
{
    char augmented_ws_path[256]; /* will append directory path */
    char *tmp_pointer;
    SpadProcess *proc;

    proc = fork_you(Die);
    child_pid = (proc == NULL ? 0 : proc->proc_id);
    switch(child_pid) {
        case -1 :
            fprintf(stderr, "Can't create a new process 
");
            exit(0);
        case 0:
            /* Dissasociate from my parents group so all my child processes */
            /* look at my terminal as the controlling terminal for the */
            /* group */

            if(setsid() < 0) {
                perror("Dissassociating from parents group");
                exit(-1);
            }

            close(ptsNum);
            /* Now reopen the server side, so that pg, su, etc. work properly */

            if((ptsNum = open(ptsPath, O_RDWR)) < 0 ) {
                perror("fork_Axiom: Failed to reopen server");
                exit(-1);
            }
            #if defined(SUN4OS5platform) || defined(HP10platform)
            ioctl(ptsNum,I_PUSH,"ptem");
            ioctl(ptsNum,I_PUSH,"ldterm");
            #endif

            /* since I am the child, I can close ptc, and dup pts for all its */
            /* standard descriptors */

            if( (dup2(ptsNum, 0) == -1) ||
                (dup2(ptsNum, 1) == -1) ||
                (dup2(ptsNum, 2) == -1) )
 CHAPTER 3. THE SMAN PROGRAM

```c
(dup2(ptsNum, 2) == -1) { perror("trying to dupe the child"); exit(-1); }
close(ptcNum);
close(ptsNum);

/* I also have to turn off echoing, since I am echoing all the */ /* input myself */
childbuf.c_lflag &= ~ECHO;
if( tcgetattr(0, TCSAFLUSH, &childbuf) == -1) {
  perror("setting the term buffer");
  exit(-1);
}
strcpy(augmented_ws_path,ws_path); /* write the name */
strcat(augmented_ws_path," "); /* space */
strcat(augmented_ws_path,ws_path); /* name again */
tmp_pointer = (char *)
  strchr(augmented_ws_path,'/'); /*pointer to last / */
  *((++tmp_pointer) = '\0';
exec_command_env(augmented_ws_path, new_envp);

  /* fprintf(stderr, "Cannot execute the %s system.\n", ws_path); */
  exit(0);
}
```

3.2.24 start_the_Axiom

```c
static void start_the_Axiom(char **envp)
{
  server_num = make_server_number();
clean_up_old_sockets();
  if (server_num == -1) {
    fprintf(stderr, "could not get an AXIOM server number\n");
    exit(-1);
  }
  if (ptyopen(&ptcNum, &ptsNum, ptcPath, ptsPath) == -1) {
    perror("start_the_Axiom: ptyopen failed");
    exit(-1);
  }
  fix_env(envp, server_num);
  fork_Axiom();
close(ptsNum);
```
3.2.25 clean_up_sockets

In order to be able to restart hyperdoc from the axiom command prompt we need to remove the socket for this server.

— sman.cleanupsockets —

static void
clean_hypertext_socket(void)
{
    char name[256];
    sprintf(name, "%s%d", MenuServerName, server_num);
    unlink(name);
}

static void
clean_up_sockets(void)
{
    char name[256];
    sprintf(name, "%s%d", SpadServer, server_num);
    unlink(name);
    sprintf(name, "%s%d", SessionServer, server_num);
    unlink(name);
    sprintf(name, "%s%d", SessionIName, server_num);
    unlink(name);
    clean_hypertext_socket();
}

— sman.readfromspadio —

3.2.26 read_from_spad_io

static void
read_from_spad_io(int ptcNum)
{
    int ret_code = 0, i=0;
    static int mes_len =0;
    ret_code = read(ptcNum, big_bad_buf, BufSize);
    if (ret_code == -1) {
        clean_up_sockets();
        exit(-1);
    }
    if (session_io == NULL) {
        if (ret_code < mes_len)
            mes_len -= ret_code;
        else {
        }
if (mes_len > 0) {
    i = mes_len;
    mes_len = 0;
} else
    i = 0;
    ret_code = write(1, big_bad_buf+i, ret_code-i);
else
    ret_code = swrite(session_io, big_bad_buf, ret_code,
                      "writing to session man");
    if (ret_code == -1) {
        perror("writing output to session manager");
        clean_up_sockets();
        exit(-1);
    }
}

3.2.27 read_from_manager

static void read_from_manager(int ptcNum)
{
    int ret_code;
    ret_code = sread(session_io, big_bad_buf, BufSize, "reading session io");
    if (ret_code == -1) {
        return;
    }
    ret_code = write(ptcNum, big_bad_buf, ret_code);
    if (ret_code == -1) {
        return;
    }
}

3.2.28 manage_spad_io

static void manage_spad_io(int ptcNum)
{
    int ret_code, i, p;
    fd_set rd;
while (1) {
    rd = socket_mask;
    FD_SET(ptcNum, &rd);
    if (session_io != NULL)
        FD_SET(session_io->socket, &rd);
    ret_code = sselect(FD_SETSIZE, &rd, 0, 0, NULL);
    if (ret_code == -1) {
        perror("Session manager select");
        clean_up_sockets();
        exit(-1);
    }
    if (FD_ISSET(ptcNum, &rd)) {
        read_from_spad_io(ptcNum);
    }
    for (i=0; i<2; i++) {
        if (server[i].socket > 0 && FD_ISSET(server[i].socket, &rd)) {
            p = accept_connection(server+i);
            switch(p) {
            case SessionIO:
                session_io = purpose_table[SessionIO];
                /* printf("connected session manager\n\r");*/
                printf("\n");
                break;
            default:
                printf("sman: Unknown connection request type: %d\n", p);
                break;
            }
        }
    }
    if (session_io != NULL && FD_ISSET(session_io->socket, &rd)) {
        read_from_manager(ptcNum);
    }
}

3.2.29 init_spad_process_list

static void
init_spad_process_list(void)
{
    spad_process_list = NULL;
}
3.2.30 print_spad_process_list

— sman.printspadprocesslist —

#include
static void
print_spad_process_list()
{
  SpadProcess *proc;
  for (proc = spad_process_list; proc != NULL; proc = proc->next)
    fprintf(stderr, "proc_id = %d, death_action = %d\n", proc->proc_id,
            proc->death_action);
} 
#endif

3.2.31 find_child

— sman.findchild —

static SpadProcess *
find_child(int proc_id)
{
  SpadProcess *proc;
  for (proc = spad_process_list; proc != NULL; proc = proc->next)
    if (proc->proc_id == proc_id) return proc;
  return NULL;
}

3.2.32 kill_all_children

— sman.killallchildren —

static void
kill_all_children(void)
{
  char name[256];
  SpadProcess *proc;

  for (proc = spad_process_list; proc != NULL; proc = proc->next) {
    kill(proc->proc_id, SIGTERM);
  }
  sprintf(name, "/tmp/hyper%d.input", server_num);
  unlink(name);
}
3.2.33 clean_up_terminal

— sman.cleanupterminal —

static void
clean_up_terminal(void)
{
    tcsetattr(0, TCSADRAIN, &oldbuf);
}

3.2.34 monitor_children

— sman.monitorchildren —

static void
monitor_children(void)
{
    int dead_baby, stat;
    SpadProcess *proc;
    while (1) {
        stat = 0;
        dead_baby = wait(&stat);
        /* Check the value of dead_baby, since wait may have returned
           a pid but subsequently we have received a signal. Yeuch!
           In order to restart hyperdoc from the axiom command prompt
           we no longer call clean_up_terminal */
        if (dead_baby == -1 && death_signal) {
            kill_all_children();
            clean_up_sockets();
            sleep(2);
            exit(0);
        }
        if (dead_baby == -1) {
            fprintf(stderr, "sman: wait returned -1\n");
            continue;
        }
        proc = find_child(dead_baby);
        if (proc == NULL) {
            /*
            fprintf(stderr, "sman: %d is not known to be a child process\n",
                dead_baby);
            */
            continue;
        }
        switch(proc->death_action) {
        /* In order to restart hyperdoc from the axiom command prompt
we no longer call clean_up_terminal. Instead we’ve added a case to just clean up the socket. */

```
case Die:
    kill_all_children();
    clean_up_sockets();
    sleep(2);
    exit(0);

case NadaDelShitsky:
    break;

case DoItAgain:
    spawn_of_hell(proc->command, DoItAgain);
    break;

case CleanHypertexSocket:
    clean_hypertex_socket();
    break;
```

3.2.35 main sman

The main procedure should return an int. We change the return value here and in src/include/sman.h1.

```
— sman.result —

return(0);

— sman.main —

```
```c
sleep(1);

if (fork_you(Die) != NULL) {
    sman_catch_signals();
    monitor_children();
    exit(0);
}

manage_spad_io(ptcNum);
if (tpd == 1) fprintf(stderr,"sman:main exit\n");
getchunk{sman.result}
}
```

### 3.2.36 sman.c

```
#define _SMAN_C

\getchunk{sman.includes}
\getchunk{sman.variables}
\getchunk{sman.processarguments}
\getchunk{sman.shouldIclef}
\getchunk{sman.inX}
\getchunk{sman.setupdefaults}
\getchunk{sman.processoptions}
\getchunk{sman.deathhandler}
\getchunk{sman.smancatchsignals}
\getchunk{sman.fixenv}
\getchunk{sman.inittermio}
\getchunk{sman.strPrefix}
\getchunk{sman.checkspadproc}
\getchunk{sman.cleanupoldsockets}
\getchunk{sman.forkyou}
\getchunk{sman.execcommandenv}
\getchunk{sman.spawnofhell}
\getchunk{sman.startthespadclient}
\getchunk{sman.startthelocalspadclient}
\getchunk{sman.startthesessionmanager}
\getchunk{sman.startthehypertex}
\getchunk{sman.starttherographics}
\getchunk{sman.forkAxiom}
\getchunk{sman.cleanupsockets}
\getchunk{sman.readfromspadio}
\getchunk{sman.readfrommanager}
\getchunk{sman.managespadio}
\getchunk{sman.initspadprocesslist}
\getchunk{sman.printspadprocesslist}
\getchunk{sman.findchild}
\getchunk{sman.killallchildren}
```
\getchunk{sman.cleanupterminal}
\getchunk{sman.monitorchildren}
\getchunk{sman.main}
Chapter 4

Support Routines

4.1 Command Completion

Hyperdoc has the ability to do command completion. The known commands are listed, one entry per line, in a file called command.list.
Chapter 5

The viewman program

— the viewman command line —

char *GraphicsProgram = "$AXIOM/lib/viewman";

— —
Chapter 6

The hypertex program

— the hypertex command line —

char *HypertexProgram = "$AXIOM/bin/hypertex -s";
Chapter 7

The clef program

— the clef command line —

char *ClefProgram = "$AXIOM/bin/clef -f $AXIOM/lib/command.list -e ";
Chapter 8

The session program

— the session manager command line —

char *SessionManagerProgram = "$AXIOM/lib/session";


8.1 session

8.1.1 includes

— ses.includes —

#include <stdlib.h>
#include <sys/time.h>
#include <stdio.h>
#include <string.h>
#include <signal.h>
#ifdef SGIplatform
#include <bstring.h>
#endif
\getchunk{include/com.h}
\getchunk{include/bsdsignal.h}
\getchunk{include/sockio-c.h1}
\getchunk{include/session.h1}


8.1.2 variables

— ses.variables —
#define BufSize 4096 /* size of communication buffer */

typedef struct sock_list { /* linked list of Sock */
    Sock Socket;
    struct sock_list *next;
} Sock_List;

Sock *spad_io = (Sock *) 0; /* to_server socket for SessionIO */
Sock *spad_server = (Sock *) 0; /* to_server socket for SpadServer */
Sock *menu_client = (Sock *) 0; /* to_client socket for MenuServerName */
Sock *active_session = (Sock *) 0; /* pointer to currently active session */
Sock_List *plSock = (Sock_List *) 0;

char big_bad_buf[BufSize]; /* big I/O buffer */
int num_active_clients = 0; /* number of InterpWindows attached */
int reading_output = 0;
fd_set session_socket_mask;

8.1.3  usr1_handler

— ses.usr1handler —

static void
usr1_handler(int sig)
{
    return;
}

8.1.4  usr2_handler

SIGUSR2 is generated by spadclients. We interpret it as an interrupt for the Lisp.

— ses.usr2handler —

static void
usr2_handler(int sig)
{
    send_signal(spad_server, SIGINT);
    return;
}
8.1.5 term_handler

— ses.termhandler —

static void
term_handler(int sig)
{
    exit(1);
}

8.1.6 pr

— ses.pr —

static void
pr()
{
    Sock_List *pSock;

    fprintf(stderr,"The socket list:
    ");
    for(pSock=plSock;pSock!=(Sock_List *)0;pSock=pSock->next){
        fprintf(stderr,"(%d,%d,%d)\t",
                pSock->Socket.pid, 2<<(pSock->Socket.socket), pSock->Socket.frame);
    }

    fprintf(stderr,"\n");
}

8.1.7 close_client

— ses.closeclient —

static void
close_client(int frame)
{
    Sock_List *pSock,*locSock;
    int socket_fd;

    /* we will check for frame equality,
     kill with send_signal,
     notify HyperTex so that it updates its list (if it’s a spadbuf),
     repair the list,
     unset the active_session,
     update num_active_clients
    */
/* first check head */
#ifdef DEBUG
fprintf(stderr,"close_client(%d)\n",frame);
#endif

if ( (plSock) && (plSock->Socket.frame == frame) ){
    socket_fd = plSock->Socket.socket;
    send_signal((Sock *)plSock, SIGTERM);
    if ( menu_client != (Sock *) 0){
        send_int(menu_client,CloseClient);
        send_int(menu_client,(*plSock).Socket.pid);
    }
#ifdef DEBUG
fprintf(stderr,"trying to clear %u\n",socket_fd);
#endif
    FD_CLR(socket_fd,&session_socket_mask);
    locSock = plSock;
    if ( (*plSock).next == (Sock_List *) 0)
        {plSock = (Sock_List *) 0;}
    else
        {plSock = plSock->next;}
    active_session = (Sock *) 0;
    num_active_clients--;
    free(locSock);
}
/* now check the rest */
else {
    for (pSock=plSock; pSock->next != (Sock_List *) 0 ; pSock=pSock->next)
        if (pSock->next->Socket.frame == frame){
            socket_fd = pSock->next->Socket.socket;
            send_signal((Sock *)pSock->next, SIGTERM);
            if ( menu_client != (Sock *) 0){
                send_int(menu_client,CloseClient);
                send_int(menu_client,(*plSock).Socket.pid);
            }
#ifdef DEBUG
            fprintf(stderr,"trying to clear %u\n",socket_fd);
#endif
            FD_CLR(socket_fd,&session_socket_mask);
            locSock = pSock->next;
            if ( pSock->next->next == (Sock_List *) 0 )
                { pSock->next= (Sock_List *) 0;}
            else
                { pSock->next = pSock->next->next;}
            num_active_clients--;
            active_session = (Sock *) 0;
            free(locSock);
            break;
        }
#ifdef DEBUG
pr();
pr();
```c
8.1.8 read_SpadServer_command

static void
read_SpadServer_command(void)
{
    int cmd, frame, num;
    cmd = get_int(spad_server);
    switch (cmd) {
    case EndOfOutput:
        if (menu_client != (Sock *)0) send_signal(menu_client, SIGUSR2);
        if (reading_output != 0) reading_output = 0;
        break;
    case QueryClients:
        /* don't count MenuServer */
        num = num_active_clients ;
        send_int(spad_server, num);
        break;
    case CloseClient:
        frame = get_int(spad_server);
        if (frame != -1) close_client(frame);
        break;
    case SendXEventToHyperTeX:
        break;
    default:
        fprintf(stderr, "session : unknown command from SpadServer %d\n", cmd);
        break;
    }
}
```

8.1.9 test_sock_for_process

```c
static int
test_sock_for_process(Sock *sock)
{
    if (sock == (Sock *)0 ) return -1;
    return kill(sock->pid, 0);
}
```
8.1.10  read_menu_client_command

--- ses.readmenuclientcommand ---

static void
read_menu_client_command(void)
{
    int cmd, frame, i, socket_fd;
    Sock_List *pSock;

    /* save it for possible clearing */
    socket_fd = menu_client->socket;

    if (test_sock_for_process(menu_client) == -1) {
        FD_CLR(socket_fd, &session_socket_mask);
        menu_client = (Sock *) 0;
        reading_output = 0;
        return;
    }
    cmd = get_int(menu_client);
    switch(cmd) {
    case -1: /* socket closed */
        FD_CLR(socket_fd, &session_socket_mask);
        menu_client = (Sock *) 0;
        reading_output = 0;
        break;
    case SwitchFrames:
        #ifdef DEBUG
        fprintf(stderr, "menu:SwitchFrames\n");
        #endif
        frame = get_int(menu_client);
        send_int(spad_server, SwitchFrames);
        send_int(spad_server, frame);
        for(i=0, pSock=plSock; pSock != (Sock_List *) 0 ; i++, pSock=pSock->next)
            if ((pSock->Socket.frame == frame)) {
                active_session = (Sock *) pSock;
                reading_output = 1;
                break;
            }
        if (i == num_active_clients) {
            /* fprintf(stderr, "Couldn't find socket for frame \%d\n", frame); */
            break;
        }
        case QuerySpad:
        #ifdef DEBUG
        fprintf(stderr, "menu:QuerySpad\n");
        #endif
        send_int(menu_client, reading_output);
        break;
        default:
            fprintf(stderr, "session : unknown command from MenuServer: \%d\n", cmd);
            menu_client = (Sock *) 0;
            break;
    }
8.1. SESSION

8.1.11 read_from_spad_io

---

ses.readfromspadio ---

static void
read_from_spad_io(void)
{
    int ret_code;
    ret_code = sread(spad_io, big_bad_buf, BufSize, "session: stdout socket");
    if (ret_code == -1) return;
    if (active_session != (Sock *) 0) {
        ret_code = swrite(active_session, big_bad_buf, ret_code, NULL);
    }
}

---

8.1.12 kill_spad

---

ses.killspad ---

static void
kill_spad(void)
{
    int i;
    Sock_List *pSock;

    send_signal(spad_server, SIGTERM);
    for (pSock = plSock, i = 0;
        i < num_active_clients) && (pSock != (Sock_List *) 0);
    i++, pSock = pSock->next) {
        if (((pSock->Socket).socket != 0)
            send_signal((Sock *) pSock, SIGTERM);
    }
    if (menu_client != (Sock *) 0) send_signal(menu_client, SIGTERM);
    exit(0);
}
8.1.13 accept_session_connection

--- ses.acceptsessionconnection ---

```c
static int
accept_session_connection(Sock *server_sock)
{
    int sock_fd, ret_code;
    Sock_List *pls;

    /* Could be three things : KillSpad MenuServer InterpWindow */

    pls = (Sock_List *) malloc(sizeof (Sock_List));
    sock_fd = accept(server_sock->socket, 0, 0);
    if (sock_fd == -1) {
        perror("session : accepting connection");
        return -1;
    }
    (pls->Socket).socket = sock_fd;
    get_socket_type((Sock *)pls);
    switch((pls->Socket).purpose) {
        case KillSpad:
            kill_spad();
            return KillSpad;
            free(pls);
        case MenuServer:
            #ifdef DEBUG
            fprintf(stderr,"session: accepted MenuServer , fd = %d\n",sock_fd);
            #endif
            menu_client = &(pls->Socket);
            FD_SET(menu_client->socket, &session_socket_mask);
            return MenuServer;
        case InterpWindow:
            #ifdef DEBUG
            fprintf(stderr,"session: accepted InterpWindow , fd = %d\n",sock_fd);
            #endif
            /* new Sock is put at the head of the list */
            if (plSock == (Sock_List *)0 ) {
                plSock = pls;
                plSock->next = (Sock_List *)0 ;
            } else{
                pls->next = plSock;
                plSock = pls;
            }
            /* we need to maintain session_socket_mask here 
            since we roll our own accept */
            FD_SET(plSock->Socket.socket, &session_socket_mask);
            send_int(spad_server, CreateFrame);
        }
    }
}
```
8.1. SESSION

```c
{
    int command = get_int(spad_server);
    /* XXX hack -- the whole protocol looks broken, we just
    try to detect losage */
    if (command != CreateFrameAnswer) {
        fprintf(stderr, "session: non-fatal, got out of sync 
"        "with Spad server\n (lost race)\n");
        /* exit(1); */
    }
    plSock->Socket.frame = get_int(spad_server);
    active_session = (Sock *)plSock;
    get_string_buf(spad_server, big_bad_buf, BufSize);
    ret_code = swrite((Sock *)plSock, big_bad_buf, strlen(big_bad_buf)+1,
                      "session: writing to InterpWindow");
    if (ret_code == -1)
        return -1;
    num_active_clients++;
#ifdef DEBUG
    pr();
#endif
    return plSock->Socket.purpose;
}
```

8.1.14 read_from_session

```c
static void
read_from_session(Sock *sock)
{
    int ret_code;
    if (sock != active_session) {
        send_int(spad_server, SwitchFrames);
        send_int(spad_server, sock->frame);
    }
    active_session = sock;
    ret_code = sread(sock, big_bad_buf, BufSize,
                     "session: reading InterpWindow");
    if (ret_code == -1) {
        active_session = (Sock *) 0;
        reading_output = 0;
        return;
    }
    ret_code = swrite(spad_io, big_bad_buf, ret_code,
                     "session: writing SessionIO");
    if (ret_code == -1) {
        active_session = (Sock *)0;
```
8.1.15 manage_sessions

--- ses.managesessions ---

```c
static void
manage_sessions(void)
{
    int ret_code;
    fd_set rd, wr, ex;
    Sock_List *pSock;

    reading_output = 0;
    while (1) {
        FD_ZERO(&rd);
        FD_ZERO(&wr);
        FD_ZERO(&ex);

        /* Allow server socket and all connections if not waiting for output
           socket_mask is maintained by libspad.a */
        #ifdef DEBUG
        fprintf(stderr,"session_socket_mask=%u ",*((long *)session_socket_mask.fds_bits));
        #endif
        rd = session_socket_mask;
        if (!reading_output) {
            rd = session_socket_mask;
        }
        /* Allow the active_session if set */
        if (active_session) FD_SET(active_session->socket, &rd);
        #ifdef DEBUG
        fprintf(stderr,"[rd=%u ",*((long *)rd.fds_bits));
        #endif
        ret_code = sselect(FD_SETSIZE, &rd, &wr, &ex, NULL);
        if (ret_code == -1) {
            break;
        }
        #ifdef DEBUG
        fprintf(stderr,"rd=%u\n",*((long *)rd.fds_bits));
        #endif

        if ((menu_client != (Sock *) 0) && FD_ISSET(menu_client->socket, &rd)) {
            /* MenuServer wants to talk */
            read_menu_client_command();
        }
```
if (FD_ISSET(spad_io->socket, &rd)) {
  /* Lisp has output */
  read_from_spad_io(); }

if (FD_ISSET(server[1].socket, &rd)) {
  /* Someone wants to connect to our server socket */
  accept_session_connection(server+1); }

for(pSock=plSock; pSock != (Sock_List *) 0 ; pSock=pSock->next) {
  if ((active_session == (Sock *)pSock || !reading_output) &&
      (pSock->Socket.socket>0 && FD_ISSET(pSock->Socket.socket, &rd)) {
    /* An InterpWindow */
    read_from_session((Sock *)pSock); }
  }

if (FD_ISSET(spad_server->socket, &rd)) {
  /* The Lisp socket */
  read_SpadServer_command(); }
}

8.1.16 main sessionmanager

— ses.main —

int
main(void)
{

#ifdef DEBUG2
  /* delay for attaching with debugger before interesting things happen */
  sleep(30);
#endif

  /* spad_server connects to Lisp server socket */
  read_SpadServer_command handles requests */
  spad_server = connect_to_local_server(SpadServer, SessionManager, Forever);
  if (spad_server == (Sock *) 0) {
    fprintf(stderr, "session: Cannot connect to AXIOM server!
"l);
    exit(0);
  }
  else {
#ifdef DEBUG
    printf(stderr, "session: connected SpadServer , fd = %d\n",
            spad_server->socket);
#endif

#endif
FD_SET(spad_server->socket, &session_socket_mask);
}

/* spad_io connects to SessionIOName server socket 
  this is Lisp std IO read_from_spad_io handles requests */
spad_io = connect_to_local_server(SessionIOName, SessionIO, Forever);
if (spad_io == (Sock *) 0) {
    fprintf(stderr, "session: Cannot connect to AXIOM IO!\n");
    exit(0);
}
else {
    #ifdef DEBUG
    fprintf(stderr,"session: connected SessionIOName , fd = %d\n",
            spad_io->socket);
    #endif
    FD_SET(spad_io->socket, &session_socket_mask);
    }
bsdSignal(SIGUSR2, usr2_handler,DontRestartSystemCalls);
bsdSignal(SIGUSR1, usr1_handler,RestartSystemCalls);
bsdSignal(SIGINT, SIG_IGN,RestartSystemCalls);
bsdSignal(SIGTERM, term_handler,RestartSystemCalls);

/* open_server opens the server socket so that we can accept connections
  we expect connections from spadbuf/spadclient(purpose:InterpWindow)
  and hypertex (MenuServer) */
if (open_server(SessionServer) == -2) {
    fprintf(stderr, "session: Cannot make server socket!\n");
    exit(-1);
}
else {
    #ifdef DEBUG
    fprintf(stderr, "session: opened SessionServer , fd = %d\n",
            server[1].socket);
    #endif
    FD_SET(server[1].socket,&session_socket_mask);
    }
manage_sessions();
return(0);
}

8.1.17 session

— session.c —

/* #define DEBUG */
#define _SESSION_C
8.1. SESSION

\getchunk{ses.includes}
\getchunk{ses.variables}
\getchunk{ses.usr1handler}
\getchunk{ses.usr2handler}
\getchunk{ses.termhandler}
\getchunk{ses.pr}
\getchunk{ses.closeclient}
\getchunk{ses.readSpadServercommand}
\getchunk{ses.testsockforprocess}
\getchunk{ses.readmenuclientcommand}
\getchunk{ses.readfromspadio}
\getchunk{ses.killspad}
\getchunk{ses.acceptsessionconnection}
\getchunk{ses.readfromsession}
\getchunk{ses.managesessions}
\getchunk{ses.main}
Chapter 9

The spadclient program

— the spadclient command line —

char *SpadClientProgram = "$AXIOM/lib/spadclient";

________

9.1 spadclient

— spadclient.c —

#define _SPADCLIENT_C

#include <stdio.h>
#include <signal.h>

#include <com.h>
#include <bsdsignal.h>
#include <bsdsignal.h1>
#include <sockio-c.h1>
#include <spadclient.h1>

Sock *sock;

static void
inter_handler(int sig)
{
    send_signal(sock, SIGUSR2);
    fflush(stderr);
}

int
main(void)
{
sock = connect_to_local_server(SessionServer, InterpWindow, Forever);
bsdSignal(SIGINT, inter_handler, RestartSystemCalls);
remote_stdio(sock);
return(0);
}
Chapter 10

The Command Completion List

— command.list —

- 
/ 
\ 
- 
~= 
- 
~= 
* 
** 
\/ 
# 
+ 
< 
<= 
= 
> 
>= 
0 
1 
abelianGroup 
abs 
absolutelyIrreducible? 
accuracyIF 
acos 
acosh 
acoshIfCan 
acosIfCan 
acot 
acoth 
acothIfCan 
acotIfCan 
acsc 
acsch 
acschIfCan
acscIfCan
aCubic
adaptive
adaptive?
adaptive3D?
addBadValue
addChild!
addData!
addField!
adddiag
addMatch
addMatchRestricted
addmod
addPoint
addPoint2
addPointLast
adjoint
airyAi
airyBi
Aleph
algDsolve
algebraic?
algebraicCoefficients?
algebraicDecompose
algebraicOf
algebraicSort
algebraicVariables
algint
algintegrate
algSplitSimple
aLinear
allRootsOf
alphabetic
alphabetic?
alphanumeric
alphanumeric?
alternating
alternatingGroup
alternative?
An
AND
And
and
anfactor
antiAssociative?
antiCommutative?
antiCommutator
anticoord
antisymmetric?
antisymmetricTensors
any
any?
append
appendPoint
apply
applyQuote
applyRules
approximants
approximate
approxNthRoot
approxSqrt
aQuadratic
aQuartic
areEquivalent?
arg1
arg2
argscript
argument
argumentList!
argumentListOf
arity
aromberg
arrayStack
asec
asech
asechIfCan
asecIfCan
asimpson
asin
asinh
asinhIfCan
asinIfCan
aspFilename
assert
assign
assoc
associatedEquations
associatedSystem
associates?
associative?
associator
associatorDependence
atan
atanh
atanhIfCan
atanIfCan
atom?
atoms
atrapezoidal
att2Result
augment
autoReduced?
axes
axesColorDefault
Bisolve
back
backOldPos
badNum
cAcsch
calcRanges
call
cap
car
cardinality
cartesian
cAsec
cAsech
cAsin
cA sinh
cAtan
cAtanh
cCos
cCosh
cCot
cCoth
cCsc
cCsch
cdr
ceiling
center
central?
certainlySubVariety?
cExp
cfirst
chainSubResultants
changeBase
changeMeasure
changeName
changeNameToObjf
changeThreshold
changeVar
changeWeightLevel
char
character?
characteristic
characteristicPolynomial
characteristicSerie
characteristicSet
charClass
charpol
charthRoot
chebyshevT
chebyshevU
check
checkCzResult
checkForZero
checkMxCDF
checkMxDF
checkPrecision
checkResult
checkRur
child
child?
children
chineseRemainder
chiSquare
chiSquare1
choose
chvar
Ci
className
clearCache
clearDenominator
clearFortranOutputStack
clearTable!
clearTheFTable
clearTheIFTTable
clearTheSymbolTable
clikeUniv
clip
clipBoolean
clipParametric
clipPointsDefault
clipSurface
clipWithRanges
cLog
close
close!
closeComponent
closed?
closedCurve
closedCurve?
cn
code
coe
coef
coefChoose
coefficient
coefficients
coerce
coerceImages
coerceListOfPairs
coerceP
coercePreimagesImages
coHeight
coelman
collect
collectQuasiMonic
collectUnder
collectUpper
color
colorDef
colorFunction
column
combineFeatureCompatibility
commaSeparate
comment
common
commonDenominator
commutative?
commutativeEquality
commutator
comp
compactFraction
companionBlocks
comparison
compBound
compdegd
compile
compiledFunction
complement
complementaryBasis
complete
completeEchelonBasis
completeEval
completeHensel
completeHermite
completeSmith
complex
complex?
complexEigenvalues
complexEigenvectors
complexElementary
complexExpand
complexForm
complexIntegrate
complexLimit
complexNormalize
complexNumeric
complexNumericIfCan
complexRoots
complexSolve
complexZeros
component
components
compose
composite
composites
computeBasis
computeCycleEntry
computeCycleLength
computeInt
computePowers
concat
concat!
cond
condition
conditionP
conditions
conditionsForIdempotents
conical
conjHerm
conjug
conjugate
conjugates
connect
connect?
cons
consnewpol
const
constant
constant?
constantCoefficientRicDE
constantIfCan
constantKernel
constantLeft
constantOperator
constantUpIfCan
constantRight
constantToUnaryFunction
constDsolve
construct
contains?
content
continue
continuedFraction
contract
contractSolve
controlPanel
convergents
convert
coord
coordinate
coordinates
copies
copy
copy!
copyInto!
corrPoly
cos
cos2sec
cosh
cosh2sech
coshIfCan
cosIfCan
cosSinInfo
cot
cot2tan
cot2trig
coth
coth2tanh
coth2trigh
cothIfCan
cotIfCan
count
countable?
countRealRoots
countRealRootsMultiple
cPower
cRationalPower
create
create3Space
createGenericMatrix
createIrreduciblePoly
createLowComplexityNormalBasis
createLowComplexityTable
createMultiplicationMatrix
createMultiplicationTable
createNormalElement
createNormalPoly
createNormalPrimitivePoly
createPrimitiveElement
createPrimitiveNormalPoly
createPrimitivePoly
createRandomElement
createThreeSpace
createZechTable
credPol
crest
critB
critBonD
critM
critMonD1
critMTonD1
critpOrder
critT
cross
crushedSet
csc
csc2sin
csch
csch2sinh
cschIfCan
cscIfCan
cSec
cSech
cSin
cSinh
csubst
cTan
cTanh
cubic
cup
currentSubProgram
curry
curryLeft
curryRight
curve
curve?
decimal
declare
declare!
decompose
decomposeFunc
decrease
decreasePrecision
deepCopy
deepestInitial
deepestTail
deepExpand
defineProperty
definingEquations
definingInequation
definingPolynomial
degree
degreePartition
degreeSubResultant
degreeSubResultantEuclidean
delay
delete
delete!
deleteProperty!
deleteRoutine!
delta
denom
denominator
denominators
denomLODE
denomRicDE
depth
dequeue
dequeue!
deref
deriv
derivationCoordinates
derivative
destruct
determinant
df2ef
df2fi
df2mf
df2st
dfList
dfRange
diag
diagonal
diagonal?
diagonalMatrix
diagonalProduct
diagonals
dictionary
diff
difference
differentialVariables
differentiate
digamma
digit
digit?
digits
dihedral
dihedralGroup
dilog
dim
dimension
dimensionOfIrreducibleRepresentation
dimensions
dimensionsOf
diophantineSystem
dioSolve
direction
directory
directProduct
directSum
discreteLog
discriminant
discriminantEuclidean
display
dispose!
distance
distdfact
distFact
distribute
div
divergence
divide
divideExponents
divideIfCan
divideIfCan!
divisor
divisorCascade
divisors
dmp2rfi
dmpToHdmp
dmpToP
dn
dom
domainOf
dominantTerm
dot
double
double?
doubleComplex?
doubleDisc
doubleRank
doubleResultant
doublyTransitive?
draw
drawComplex
drawComplexVectorField
drawCurves
drawStyle
drawToScale
droot
duplicates
duplicates?
e
e01baf
e01bef
e01bff
e01bgf
e01bhf
e01daf
e01saf
e01sbf
e01sef
e01sff
e02adf
e02aef
e02agf
e02ahf
e02ajf
e02akf
e02baf
e02bbf
e02bcf
e02bdf
e02bef
e02daf
e02dcf
e02ddf
e02def
e02dff
e02gaf
e02zaf
e04dgf
e04fdf
e04gcf
e04jaf
e04mbf
e04naf
e04ucf
e04ycf
edf2df
edf2ef
edf2efi
edf2fi
ef2edf
Ei
eigenMatrix
eigenvalues
eigenvector
expandLog
expandPower
expandTrigProducts
expenseOfEvaluation
expenseOfEvaluationIF
expextendedit
expIfCan
expint
expintegrate
expintfldpoly
explicitEntries?
explicitlyEmpty?
explicitlyFinite?
explimitedint
explogs2trigs
exponent
exponential
exponential1
exponential1Order
exponents
expPot
expr
expressIdealMember
exprHasAlgebraicWeight
exprHasLogarithmicWeights
exprHasWeightCosWXorSinWX
exprToGenUPS
exprToUPS
exprToXXP
expt
exptMod
exQuo
exquo
extend
extendedEuclidean
extendedint
extendedIntegrate
extendedResultant
extendedSubResultantGcd
extendIfCan
extension
extensionDegree
exteriorDifferential
external?
externalList
extract!
extractBottom!
extractClosed
extractIfCan
extractIndex
extractPoint
extractProperty
extractSplittingLeaf
extractTop!
eyeDistance
F
f01brf
f01bsf
f01maf
f01mcf
f01qcf
f01qdf
f01qef
f01rcf
f01rdf
f01ref
f02aaf
f02abf
f02adf
f02aef
f02aff
f02agf
f02ajf
f02akf
f02awf
f02axf
f02bbf
f02bjf
f02fjf
f02wef
f02xef
f04adf
f04arf
f04asf
f04atf
f04axf
f04faf
f04jgf
f04maf
f04mbf
f04mcf
f04qaf
f07adf
f07aef
f07fdf
f07fef
f2df
F2FG
f2st
factor
factor1
factorAndSplit
factorByRecursion
factorFraction
factorGroebnerBasis
factorial
factorials
factorList
factorOfDegree
factorPolynomial
factors
factorSet
factorSFBRlcUnit
factorsOfCyclicGroupSize
factorsOfDegree
factorSquareFree
factorSquareFreeByRecursion
factorSquareFreePolynomial
failed
failed?
false
ffactor
FG2F
fg1m1fCan
fi2df
fibonacci
field
fields
figureUnits
filename
fill!
fillPascalTriangle
filterUntil
filterWhile
find
findCycle
finite?
finiteBasis
finiteBound
fintegrate
first
firstDenom
firstNumer
firstSubsetGray
firstUncouplingMatrix
fixedDivisor
fixedPoint
fixedPointExquo
fixedPoints
fixPredicate
flagFactor
flatten
flexible?
flexibleArray
float
float?
floatlist
floatlist?
floor
fmecg
forLoop
FormatArabic
generalTwoFactor
generate
generateIrredPoly
generator
generators
generic
generic?
genericLeftDiscriminant
genericLeftMinimalPolynomial
genericLeftNorm
genericLeftTrace
genericLeftTraceForm
genericPosition
genericRightDiscriminant
genericRightMinimalPolynomial
genericRightNorm
genericRightTrace
genericRightTraceForm
genus
geometric
getBadValues
getButtonValue
getCode
getCurve
getDatabase
getExplanations
getGoodPrime
getGraph
gethi
getlo
getMatch
getMeasure
getMultiplicationMatrix
getMultiplicationTable
getOrder
getPickedPoints
getRef
getStream
getVariableOrder
getZechTable
GF2FG
goodnessOfFit
goodPoint
GospersMethod
goto
gradient
graeffe
gramschmidt
graphCurves
graphImage
graphs
graphState
graphStates
green
hyperelliptic
hypergeometric0F1
iCompose
id
ideal
idealiser
idealiserMatrix
idealSimplify
identification
identity
identityMatrix
identitySquareMatrix
iExquo
iflist2Result
iFTable
ignore?
iiabs
iiacos
iiacosinh
iiacot
iiacoth
iiacsc
iiacsch
iiasec
iiasech
iiasin
iiasinh
iiatan
iiatanh
iibinom
iicos
iicosh
iicot
iicoth
iicsc
iicsch
iidprod
iidsum
iiexp
iifact
iiGamma
iilog
iiperm
iipow
iisec
iisech
iiisin
iiisinh
iiisqrt2
iiisqrt3
iiitan
iiitanh
imag
imagE
imagI
imagi
imaginary
imagJ
imagj
imagK
imagk
implies
in?
in
crc
increase
increasePrecision
increment
incrementBy
incrementKthElement
index
index?
indices
indiceSubResultant
indiceSubResultantEuclidean
indicialEquation
indicialEquationAtInfinity
indicialEquations
inf
infieldint
infieldIntegrate
infinite?
infiniteProduct
infinity
infinityNorm
infix
infix?
inLex?
inRittWu?
inGroundField?
inHallBasis?
init
initial
initializeGroupForWordProblem
initiallyReduce
initiallyReduced?
initials
initTable!
innerEigenvectors
innerint
innerSolve
innerSolve1
input
inR?
in Radical?
inrootof
insert
insert!
insertBottom!
insertionSort!
insertMatch
insertRoot!
insertTop!
inspect
int
int?
intChoose
 intcompBasis
integer
integer?
integerBound
integerIfCan
integers
integral
integral?
integralAtInfinity?
integralBasis
integralBasisAtInfinity
integralCoordinates
integralDerivationMatrix
integralLastSubResultant
integralMatrix
integralMatrixAtInfinity
integralRepresents
integrate
intensity
intermediateResultsIF
internal?
internalAugment
internalDecompose
internalInfRittWu?
internalIntegrate
internalIntegrate0
internalLastSubResultant
internalSubPolSet?
internalSubQuasiComponent?
internalZeroSetSplit
interpolate
interpret
interpretString
interReduce
intersect
interval
intlist
intlist?
intPatternMatch
inv
inverse
inverseColeman
inverseIntegralMatrix
inverseIntegralMatrixAtInfinity
inverseLaplace
lambda
lambert
laplace
laplacian
largest
last
lastSubResultant
lastSubResultantElseSplit
lastSubResultantEuclidean
latex
laurent
laurentIfCan
laurentRep
Lazard
Lazard2
LazardQuotient
LazardQuotient2
lazy?
lazyEvaluate
lazyGintegrate
lazyIntegrate
lazyIrreducibleFactors
lazyPquo
lazyPrem
lazyPremWithDefault
lazyPseudoDivide
lazyPseudoQuotient
lazyPseudoRemainder
lazyResidueClass
lazyVariations
lcm
ldf2lst
ldf2vmf
LE
leader
leadingBasisTerm
leadingCoefficient
leadingCoefficientRicDE
leadingExponent
leadingIdeal
leadingIndex
leadingMonomial
leadingSupport
leadingTerm
leaf?
leastAffineMultiple
leastMonomial
leastPower
leaves
left
leftAlternative?
leftCharacteristicPolynomial
leftDiscriminant
leftDivide
limit
limitedInt
limitedIntegrate
limitPlus
linear
linear?
linearAssociatedExp
linearAssociatedLog
linearAssociatedOrder
linearDependence
linearDependenceOverZ
linearlyDependent?
linearlyDependentOverZ?
linearMatrix
linearPart
linearPolynomials
linears
lineColorDefault
linGenPos
linkToFortran
linSolve
lintgcd
list
list?
listBranches
listConjugateBases
listexp
listLoops
listOfLists
listOfMonoms
listOfTerms
listRepresentation
lists
listYoungTableaus
lllip
lllp
l1prop
lo
localAbs
localIntegralBasis
localReal?
localUnquote
LOD02FUN
log
log10
log2
logGamma
logical?
logIfCan
logpart
lookup
loopPoints
low
lowerCase
makeViewport3D
makeYoungTableau
makingStats?
mantissa
map
map!
mapBivariate
mapCoef
mapdiv
mapDown!
mapExpon
mapExponents
mapGen
mapMatrixIfCan
mapmult
mapSolve
mapUnivariate
mapUnivariateIfCan
mapUp!
mask
mat
match
match?
mathieu11
mathieu12
mathieu22
mathieu23
mathieu24
matrix
matrixConcat3D
matrixDimensions
matrixGcd
max
maxColIndex
maxdeg
maximumExponent
maxIndex
maxint
maxPoints
maxPoints3D
maxrank
maxrow
maxRowIndex
mdeg
measure
measure2Result
meatAxe
medialSet
member?
members
merge
merge!
mergeDifference
mergeFactors
monicModulo
monicRightDivide
monicRightFactorIfCan
monom
monomial
monomial?
monomialIntegrate
monomialIntPoly
monomials
monomRDE
monomRDEsys
more?
moreAlgebraic?
morphism
move
movedPoints
mpsode
mr
mulmod
multiEuclidean
multiEuclideanTree
multinomial
multiple
multiple?
multiplyCoefficients
multiplyExponents
multisect
multiset
multivariate
multMonom
musserTrials
mvar
myDegree
nagCosInt
nagDAiryAi
nagDAiryBi
nagDFT
nagEigenvalues
nagEigenVectors
nagEllipticIntegralRC
nagEllipticIntegralRD
nagEllipticIntegralRF
nagEllipticIntegralRJ
nagErf
nagErfC
nagExpInt
nagFresnelC
nagFresnelS
nagHankelH1
nagHankelH2
nagHermitianDFT
nagHermitianInverseDFT
nagIncompleteGammaP
nagIncompleteGammaQ
normal?
normal01
normalDenom
normalDeriv
normalElement
normalForm
normalise
normalize
normalizeAtInfinity
normalized?
normalizedAssociate
normalizedDivide
normalizeIfCan
normDeriv2
normFactors
normInvertible?
NUT
Not
not
notelem
npcoef
nrows
nsqfree
nthCoeff
nthExpon
nthExponent
nthFactor
nthFlag
nthFractionalTerm
nthr
nthRoot
nthRootIfCan
Nul
null
null?
nunary
nullary?
nunarity
nullspace
number?
numberOfChildren
numberOfComponents
numberOfComposites
numberOfComputedEntries
numberOfCycles
numberOfDivisors
numberOfFactors
numberOfFractionalTerms
numberOfHues
numberOfImproperPartitions
numberOfIrreduciblePoly
numberOfMonomials
numberOfNormalPoly
numberOfOperations
numberOfPrimitivePoly
numberOfVariables
numer
numerator
numerator
numeric
numericalIntegration
numericalOptimization
numericIfCan
numFunEvals
numFunEvals3D
obj
objectOf
objects
oblateSpheroidal
ocf2ocdf
octon
odd?
oddInfiniteProduct
oddintegers
oddlambert
ode
ode1
ode2
ODESolve
OMbindTCP
OMclose
OMcloseConn
OMconnectTCP
OMconnInDevice
OMconnOutDevice
OMencodingBinary
OMencodingSGML
OMencodingUnknown
OMencodingXML
omError
OMgetApp
OMgetAtp
OMgetAttr
OMgetBind
OMgetBVar
OMgetEndApp
OMgetEndAtp
OMgetEndAttr
OMgetEndBind
OMgetEndBVar
OMgetEndError
OMgetEndObject
OMgetError
OMgetFloat
OMgetInteger
OMgetObject
OMgetString
OMgetSymbol

OMgetType
OMgetVariable
OMlistCDs
OMlistSymbols
OMmakeConn
OMopenFile
OMopenString
OMParseError?
OMputApp
OMputAtp
OMputAttr
OMputBind
OMputBVar
OMputEndApp
OMputEndAtp
OMputEndAttr
OMputEndBind
OMputEndBVar
OMputEndError
OMputEndObject
OMputError
OMputFloat
OMputInteger
OMputObject
OMputString
OMputSymbol
OMputVariable
OMread
OMReadError?
OMreadFile
OMreadStr
OMreceive
OMsend
OMserve
OMsetEncoding
OMsupportsCD?
OMsupportsSymbol?
OMunhandledSymbol
OMUnknownCD?
OMUnknownSymbol?
OMwrite
one?
oneDimensionalArray
op
open
open?
operation
operator
operators
opeval
optAttributes
optimize
option
option?
optional
optional?
options
optpair
OR
Or
or
orbit
orbits
ord
order
orthonormalBasis
outerProduct
outlineRender
output
outputArgs
outputAsFortran
outputAsScript
outputAsTex
outputFixed
outputFloating
outputForm
outputGeneral
outputList
outputMeasure
outputSpacing
over
overbar
overlabel
overlap
overset?
pack!
packageCall
packHS
pade
padecf
padicallyExpand
padicFraction
pair?
palgextint
palgextint0
palginfeldint
palgint
palgint0
palgintegrate
palglimint
palglimint0
palgLODE
palgLODE0
palgRDE
palgRDE0
parabolic
parabolicCylindrical
paraboloidal
parametersOf
parametric?
ParCond
ParCondList
paren
parent
partialDenominators
partialFraction
partialNumerator
partialQuotients
particularSolution
partition
partitions
parts
pascalTriangle
pastel
pattern
patternMatch
patternMatchTimes
patternVariable
pdct
PDESolve
pdf2df
pdf2ef
perfectNthPower?
perfectNthRoot
perfectSqrt
perfectSquare?
permanent
permutation
permutationGroup
permutationRepresentation
permutations
perspective
phiCoord
pHS
physicalLength
physicalLength!
pi
pile
plenaryPower
pleskenSplit
plot
plotPolar
plus
plus!
plusInfinity
pmComplexIntegrate
pmIntegrate
po
point
point?
pointColor
pointColorDefault
preprocess
presub
presuper
previous
prevPrime
primaryDecomp
prime
prime?
primeFactor
primeFrobenius
primes
primextendint
primextendintfrac
primintegrate
primintfldpoly
primitive?
primitiveElement
primitiveMonomials
primitivePart
primitivePart!
primlimintfrac
primlimitedint
primPartElseUnitCanonical
primPartElseUnitCanonical!
prinb
principal?
principalIdeal
prindINFO
prinpolINFO
prinshINFO
print
printCode
printHeader
printInfo
printInfo!
printingInfo?
printStatement
printStats!
printTypes
probablyZeroDim?
problemPoints
processTemplate
prod
product
prolateSpheroidal
prologue
properties
property
pseudoDivide
pseudoQuotient
pseudoRemainder
psolve
ptFunc
pToDmp
radix
radPoly
raisePolynomial
ramified?
ramifiedAtInfinity?
ran
randnum
random
randomLC
randomR
range
rangeIsFinite
rangePascalTriangle
ranges
rank
rarrow
ratDenom
ratDsolve
rational
rational?
rationalApproximation
rationalFunction
rationalIfCan
rationalPoint?
rationalPoints
rationalPower
ratpart
ratPoly
ravel
rCoord
rdHack1
rdregime
read
read!
readable?
readIfCan!
readLine!
readLineIfCan!
real
real?
realEigenvalues
realEigenvectors
realElementary
realRoots
realSolve
realZeros
recip
reciprocalPolynomial
recolor
recoverAfterFail
rectangularMatrix
recur
red
redmat
reorder
repeating
repeating?
repeatUntilLoop
replace
replaceKthElement
representationType
represents
repSq
reseed
reset
reset!
resetAttributeButtons
resetBadValues
resetNew
resetVariableOrder
reshape
resize
rest
restorePrecision
result
resultant
resultantEuclidean
resultantEuclideannaif
resultanntnaif
resultantReduit
resultantReduitEuclidean
retract
retractable?
retractIfCan
returns
returnType!
returnTypeOf
reverse
reverse!
reverseLex
revert
rewriteIdealWithHeadRemainder
rewriteIdealWithQuasiMonicGenerators
rewriteIdealWithRemainder
rewriteSetByReducingWithParticularGenerators
rewriteSetWithReduction
RFZUTS
rhs
ricDsolve
ridHack1
right
rightAlternative?
rightCharacteristicPolynomial
rightDiscriminant
rightDivide
rightExactQuotient
rightExtendedGcd
rightFactorCandidate
rightFactorIfCan
rightGcd
rightLcm
rightMinimalPolynomial
rightMult
rightNorm
rightOne
rightPower
rightQuotient
rightRank
rightRankPolynomial
rightRecip
rightRegularRepresentation
rightRemainder
rightScalarTimes!
rightTrace
rightTraceMatrix
rightTrim
rightUnit
rightUnits
rightZero
rischDE
rischDEsys
rischNormalize
RittWuCompare
rk4
rk4a
rk4f
rk4qc
roman
romberg
rombergo
root
root?
rootBound
rootKerSimp
rootNormalize
root0f
root0fIrreduciblePoly
rootPoly
rootPower
rootProduct
rootRadius
rootSimp
roots0f
rootSplit
rotate
rotate!
rotatex
rotatex
rotatey
rotatey
rotatez
rotatez
roughBase?
roughBasicSet
roughEqualIdeals?
roughSubIdeal?
roughUnitIdeal?
round
routines
rov
row
rowEch
rowEchelon
rowEchelonLocal
rowEchLocal
rquo
rroot
rspace
rst
rubiksGroup
rule
rules
ruleset
rur
s01eaf
s13aaf
s13acf
s13adf
s14aaf
s14abf
s14baf
s15adf
s15aef
s17acf
s17adf
s17aef
s17aff
s17agf
s17ahf
s17ajf
s17akf
s17dcf
s17def
s17dgf
s17dhf
s18acf
s18adf
s18aeaf
s18aff
s18dcf
s18def
s19aaf
s19abf
s19abf
s19acf
s19adf
s20acf
s20adf
s21baf
s21bbf
a21bcf
a21bdf
safeCeiling
safeFloor
safetyMargin
sample
satisfy?
saturate
save
say
sayLength
scalarMatrix
scalarTypeOf
scale
scaleRoots
scan
ScanArabic
ScanFloatIgnoreSpaces
ScanFloatIgnoreSpacesIfCan
scanOneDimSubspaces
ScanRoman
schema
schwerpunkt
screenResolution
screenResolution3D
script
scripted?
scripts
sdf21st
se2rfi
search
sec
sec2cos
sech
sech2cosh
sechIfCan
secIfCan
second
seed
SEGMENT
segment
select
select!
selectAndPolynomials
selectFiniteRoutines
selectfirst
selectIntegrationRoutines
selectMultiDimensionalRoutines
selectNonFiniteRoutines
selectODEIVPRoutines
selectOptimizationRoutines
selectOrPolynomials
selectPDERoutines
selectPolynomials
selectsecond
selectSumOfSquaresRoutines
semicolonSeparate
semiDegreeSubResultantEuclidean
semiDiscriminantEuclidean
semiIndiceSubResultantEuclidean
semiLastSubResultantEuclidean
semiResultantEuclidean1
semiResultantEuclidean2
semiResultantEuclideannaif
semiResultantReduitEuclidean
semiSubResultantGcdEuclidean1
semiSubResultantGcdEuclidean2
separant
separate
separateDegrees
separateFactors
sequences
series
seriesSolve
seriesToOutputForm
set
setAdaptive
setAdaptive3D
setAttributeButtonStep
setButtonValue
setchildren!
setClipValue
setClosed
setColumn!
setCondition!
setDifference
setelt
setelt!
setEmpty!
setEpilogue!
setErrorBound
setFieldInfo
setfirst!
setFormula!
setImagSteps
setIntersection
setLabelValue
setlast!
setleaves!
setleft!
setLegalFortranSourceExtensions
setMaxPoints
setMaxPoints3D
setMinPoints
setMinPoints3D
setnext!
setOfMinN
setOrder
setPoly
setPosition
setPredicates
setprevious!
setPrologue!
setProperties
setProperty
setRealSteps
setref
setrest!
setright!
setRow!
setScreenResolution
setScreenResolution3D
setStatus
setStatus!
setsubMatrix!
setTex!
setTopPredicate
setUnion
setValue!
setvalue!
setVariableOrder
SFunction
sh
shade
shallowCopy
shallowExpand
shanksDiscLogAlgorithm
shellSort
shift
shiftLeft
shiftRight
shiftRoots
show
showAll?
showAllElements
showArrayValues
showAttributes
showClipRegion
showFortranOutputStack
showIntensityFunctions
showRegion
showScalarValues
showTheFTable
showTheIPTable
showTheRoutinesTable
showTheSymbolTable
showTheSymbolTable
showTypeInOutput
 shrinkable
shuffle
shufflein
Si
sign
signAround
simpleBounds?
simplify
simplifyExp
simplifyLog
simplifyPower
simpson
simpsono
sin
sin?
sin2csc
sincos
singleFactorBound
singRicDE
singular?
singularAtInfinity?
singularitiesOf
sinh
sinh2csc
sinhcosh
sinhIfCan
sinIfCan
size
size?
sizeLess?
sizeMultiplication
sizePascalTriangle
skewSFunction
slash
slex
smith
sn
sncndn
socf2socdf
solid
solid?
solve
solve1
solveid
solveInField
solveLinear
solveLinearlyOverQ
solveLinearPolynomialEquation
solveLinearPolynomialEquationByFractions
solveLinearPolynomialEquationByRecursion
solveRetract
someBasis
sort
sort!
sortConstraints
sorted?
space
sparsityIF
specialTrigs
spherical
split
split!
splitConstant
splitDenominator
splitLinear
splitNodeOf!
splitSquarefree
sPol
sqfree
sqfrFactor
sqrt
square?
squareFree
squareFreeFactors
squareFreeLexTriangular
squareFreePart
squareFreePolynomial
squareFreePrim
squareMatrix
squareTop
stack
standardBasisOfCyclicSubmodule
start!
startPolynomial
startStats!
startTable!
startTableGcd!
startTableInvSet!
status
stFunc1
stFunc2
stFuncN
stiffnessAndStabilityFactor
stiffnessAndStabilityOfODEIF
stirling1
stirling2
stop
stop!
stopMusserTrials
stopTable!
stopTableGcd!
stopTableInvSet!
stoseIntegralLastSubResultant
stoseInternallLastSubResultant
stoseInvertible?
stoseInvertible?reg
stoseInvertibleSet
stoseInvertibleSetreg
stoseInvertibleSetsqfreg
stoseInvertible?sqfreg
stoseLastSubResultant
stosePrepareSubResAlgo
stoseSquareFreePart
string
string?
stripCommentsAndBlanks
strongGenerators
stronglyReduce
stronglyReduced?
structuralConstants
sts2stst
SturmHabicht
SturmHabichtCoefficients
SturmHabichtMultiple
SturmHabichtSequence
sturmSequence
sturmVariationsOf
style
sub
subCase?
subHeight
subMatrix
submod
subNode?
subNodeOf?
subPolSet?
subQuasiComponent?
subResultantChain
subResultantGcd
subResultantGcdEuclidean
subResultantsChain
subresultantSequence
subresultantVector
subscript
subscriptedVariables
subSet
subset?
subspace
subst
substitute
substring?
subtractIfCan
subTriSet?
suchThat
suffix?
sup
summation
sumOfDivisors
sumOfKthPowerDivisors
sumOfSquares
sumSquares
sup
supDimElseRittWu?
super
superHeight
superscript
supersub
tex
thetaCoord
third
timer
times
times!
title
top
top!
topFortranOutputStack
topPredicate
toroidal
torsion?
torsionIfCan
toScale
toseInvertible?
toseInvertibleSet
toseLastSubResultant
toseSquareFreePart
totalDegree
totalDifferential
totalfract
totalGroebner
totalLex
totolex
tower
trace
trace2PowMod
traceMatrix
tracePowMod
trailingCoefficient
tRange
transcendenceDegree
transcendent?
transcendentalDecompose
transform
translate
transpose
trapezoidal
trapezoidalo
traverse
tree
triangSolve
triangular?
triangularSystems
triangulate
trigs
trigs2explogs
trim
trivialIdeal?
true
trueEqual
trunc
truncate
tryFunctionalDecomposition
tryFunctionalDecomposition?
tube
tubePlot
tubePoints
tubePointsDefault
tubeRadius
tubeRadiusDefault
tValues
twist
twoFactor
typeList
typeLists
unary?
unaryFunction
uncouplingMatrices
unexpand
uniform
uniform01
union
uniqueID
unit
unit?
unitCanonical
unitNormal
unitNormalize
units
unitsColorDefault
unitVector
univariate
univariate?
univariatePolynomial
univariatePolynomials
univariatePolynomialsGcds
univariateSolve
univcase
universe
unmakeSUP
unparse
unprotectedRemoveRedundantFactors
unrankImproperPartitions0
unrankImproperPartitions1
unravel
untab
UnVectorise
unvectorise
UP2ifCan
UP2UTS
updatD
update
upDateBranches
updateStatus!
updatF
upperCase
upperCase!
upperCase?
UpTriBddDenomInv
useEisensteinCriterion
useEisensteinCriterion?
useNagFunctions
userOrdered?
useSingleFactorBound
useSingleFactorBound?
usingTable?
UT52UP
validExponential
value
var1Steps
var1StepsDefault
var2Steps
var2StepsDefault
variable
variables
variationOfParameters
vark
varList
varselect
vconcat
vector
Vectorise
vectorise
vedf2vef
vertConcat
viewDefaults
viewDeltaXDefault
viewDeltaYDefault
viewPhiDefault
viewpoint
viewport2D
viewport3D
viewPosDefault
viewSizeDefault
viewThetaDefault
viewWriteAvailable
viewWriteDefault
viewZoomDefault
virtualDegree
void
vput!
vspace
vstart!
walkTree
weakBiRank
weierstrass
weight
weighted
weights
whatInfinity
whileLoop
wholePart
wholeRadix
wholeRagits
width
withPredicates
wordInGenerators
wordInStrongGenerators
wordsForStrongGenerators
wreath
writable?
write
write!
writeLine!
wronskianMatrix
wrregime
xCoord
xn
xor
xRange
Y
yCoord
yCoordinates
yellow
youngGroup
yRange
zag
zCoord
zero
zero?
zeroDim?
zeroDimensional?
zeroDimPrimary?
zeroDimPrime?
zeroMatrix
zeroOf
zeroSetSplit
zeroSetSplitIntoTriangularSystems
zerosOf
zeroSquareMatrix
zeroVector
zoom
zRange
AbelianGroup
AbelianMonoid
AbelianMonoidRing
AbelianSemiGroup
Aggregate
Algebra
AlgebraicallyClosedField
AlgebraicallyClosedFunctionSpace
ArcHyperbolicFunctionCategory
ArcTrigonometricFunctionCategory
AssociationListAggregate
AttributeRegistry
BagAggregate
BasicType
BiModule
BinaryRecursiveAggregate
BinaryTreeCategory
BitAggregate
CachableSet
CancellationAbelianMonoid
CharacteristicNonZero
CharacteristicZero
CoercibleTo
Collection
CombinatorialFunctionCategory
CombinatorialOpsCategory
CommutativeRing
ComplexCategory
ConvertibleTo
DequeueAggregate
Dictionary
DictionaryOperations
DifferentialExtension
DifferentialPolynomialCategory
DifferentialRing
DifferentialVariableCategory
DirectProductCategory
DivisionRing
DoublyLinkedAggregate
ElementaryFunctionCategory
Eltable
EltableAggregate
EntireRing
EuclideanDomain
Evalable
ExpressionSpace
ExtensibleLinearAggregate
ExtensionField
Field
FieldOfPrimeCharacteristic
FileCategory
FileNameCategory
Finite
FiniteAbelianMonoidRing
FiniteAlgebraicExtensionField
FiniteDivisorCategory
FiniteFieldCategory
FiniteLinearAggregate
FiniteRankAlgebra
FiniteRankNonAssociativeAlgebra
FiniteSetAggregate
FloatingPointSystem
FortranFunctionCategory
FortranMachineTypeCategory
FortranMatrixCategory
NumericalIntegrationCategory
NumericalOptimizationCategory
OctonionCategory
OneDimensionalArrayAggregate
OpenMath
OrderedAbelianGroup
OrderedAbelianMonoid
OrderedAbelianMonoidSup
OrderedAbelianSemiGroup
OrderedCancellationAbelianMonoid
OrderedFinite
OrderedIntegralDomain
OrderedMonoid
OrderedMultisetAggregate
OrderedRing
OrderedSet
OrdinaryDifferentialEquationsSolverCategory
PAdicIntegerCategory
PartialDifferentialEquationsSolverCategory
PartialDifferentialRing
PartialTranscendentalFunctions
Patternable
PatternMatchable
PermutationCategory
PlottablePlaneCurveCategory
PlottableSpaceCurveCategory
PointCategory
PolynomialCategory
PolynomialFactorizationExplicit
PolynomialSetCategory
PowerSeriesCategory
PrimitiveFunctionCategory
PrincipalIdealDomain
PriorityQueueAggregate
QuaternionCategory
QueueAggregate
QuotientFieldCategory
RadicalCategory
RealClosedField
RealConstant
RealNumberSystem
RealRootCharacterizationCategory
RectangularMatrixCategory
RecursiveAggregate
RecursivePolynomialCategory
RegularTriangularSetCategory
RetractableTo
RightModule
Ring
Rng
SegmentCategory
SegmentExpansionCategory
SemiGroup
SetAggregate
SetCategory
SExpressionCategory
SpecialFunctionCategory
SquareFreeNormalizedTriangularSetCategory
SquareFreeRegularTriangularSetCategory
SquareMatrixCategory
StackAggregate
StepThrough
StreamAggregate
StringAggregate
StringCategory
TableAggregate
ThreeSpaceCategory
TranscendentalFunctionCategory
TriangularSetCategory
TrigonometricFunctionCategory
TwoDimensionalArrayCategory
Type
UnaryRecursiveAggregate
UniqueFactorizationDomain
UnivariateLaurentSeriesCategory
UnivariateLaurentSeriesConstructorCategory
UnivariatePolynomialCategory
UnivariatePowerSeriesCategory
UnivariatePuiseuxSeriesCategory
UnivariatePuiseuxSeriesConstructorCategory
UnivariateSkewPolynomialCategory
UnivariateTaylorSeriesCategory
VectorCategory
VectorSpace
XAlgebra
XFreeAlgebra
XPolynomialsCat
AlgebraGivenByStructuralConstants
AlgebraicFunctionField
AlgebraicNumber
AnonymousFunction
AntiSymm
Any
ArrayStack
Asp1
Asp10
Asp12
Asp19
Asp20
Asp24
Asp27
Asp28
Asp29
Asp30
Asp31
Asp33
Asp34
Asp35
Asp4
Asp41
Asp42
Asp49
Asp50
Asp55
Asp6
Asp7
Asp73
Asp74
Asp77
Asp78
Asp8
Asp80
Asp9
AssociatedJordanAlgebra
AssociatedLieAlgebra
AssociationList
AttributeButtons
Automorphism
BalancedBinaryTree
BalancedPAdicInteger
BalancedPAdicRational
BasicFunctions
BasicOperator
BinaryExpansion
BinaryFile
BinarySearchTree
BinaryTournament
BinaryTree
Bits
Boolean
CardinalNumber
CartesianTensor
Character
CharacterClass
CliffordAlgebra
Color
Commutator
Complex
ContinuedFraction
d01ajfAnnaType
d01akfAnnaType
d01alfAnnaType
d01amfAnnaType
d01anfAnnaType
d01apfAnnaType
d01aqfAnnaType
d01asfAnnaType
d01fcfAnnaType
d01gbfAnnaType
d01TransformFunctionType
d02bbfAnnaType
d02bhfAnnaType
FortranScalarType
FortranTemplate
FortranType
FourierComponent
FourierSeries
Fraction
FractionalIdeal
FramedModule
FreeAbelianGroup
FreeAbelianMonoid
FreeGroup
FreeModule
FreeModule1
FreeMonoid
FreeNilpotentLie
FullPartialFractionExpansion
FunctionCalled
GeneralDistributedMultivariatePolynomial
GeneralModulePolynomial
GeneralPolynomialSet
GeneralSparseTable
GeneralTriangularSet
GeneralUnivariatePowerSeries
GenericNonAssociativeAlgebra
GraphImage
HashTable
Heap
HexadecimalExpansion
HomogeneousDirectProduct
HomogeneousDistributedMultivariatePolynomial
HyperellipticFiniteDivisor
IndexCard
IndexedBits
IndexedDirectProductAbelianGroup
IndexedDirectProductAbelianMonoid
IndexedDirectProductObject
IndexedDirectProductOrderedAbelianMonoid
IndexedDirectProductOrderedAbelianMonoidSup
IndexedExponents
IndexedFlexibleArray
IndexedList
IndexedMatrix
IndexedOneDimensionalArray
IndexedString
IndexedTwoDimensionalArray
IndexedVector
InfiniteTuple
InnerAlgebraicNumber
InnerFiniteField
InnerFreeAbelianMonoid
InnerIndexedTwoDimensionalArray
InnerPAdicInteger
InnerPrimeField
InnerSparseUnivariatePowerSeries
ModuleOperator
MoebiusTransform
MonoidRing
Multiset
MultivariatePolynomial
NagDiscreteFourierTransformInterfacePackage
NagEigenInterfacePackage
NagOptimisationInterfacePackage
NagQuadratureInterfacePackage
NagResultChecks
NagSpecialFunctionsInterfacePackage
NewSparseMultivariatePolynomial
NewSparseUnivariatePolynomial
None
NonNegativeInteger
NumericalIntegrationProblem
NumericalODEProblem
NumericalOptimizationProblem
NumericalPDEProblem
Octonion
ODEIntensityFunctionsTable
OneDimensionalArray
OnePointCompletion
OpenMathConnection
OpenMathDevice
OpenMathEncoding
OpenMathError
OpenMathErrorKind
Operator
OppositeMonogenicLinearOperator
OrderedCompletion
OrderedDirectProduct
OrderedFreeMonoid
OrderedVariableList
OrderlyDifferentialPolynomial
OrderlyDifferentialVariable
OrdinaryDifferentialRing
OrdinaryWeightedPolynomials
OrdSetInts
OutputForm
PackedHermitianSequence
PAthicInteger
PAthicRational
PAthicRationalConstructor
Palette
ParametricPlaneCurve
ParametricSpaceCurve
ParametricSurface
PartialFraction
Partition
Pattern
PatternMatchListResult
PatternMatchResult
PendantTree
Permutation
PermutationGroup
Pi
PlaneAlgebraicCurvePlot
Plot
Plot3D
PoincareBirkhoffWittLyndonBasis
Point
Polynomial
PolynomialIdeals
PolynomialRing
PositiveInteger
PrimeField
PrimitiveArray
Product
QuadraticForm
QuasiAlgebraicSet
Quaternion
QueryEquation
Queue
RadicalFunctionField
RadixExpansion
RealClosure
Record
RectangularMatrix
Reference
RegularChain
RegularTriangularSet
RenderTools
ResidueRing
Result
RewriteRule
RightOpenIntervalRootCharacterization
RomanNumeral
RoutinesTable
RuleCalled
Ruleset
ScriptFormulaFormat
Segment
SegmentBinding
SequentialDifferentialPolynomial
SequentialDifferentialVariable
Set
SetOfMIntegersInOneToN
SExpression
SExpressionOf
SimpleAlgebraicExtension
SimpleFortranProgram
SingleInteger
SingletonAsOrderedSet
SparseMultivariatePolynomial
SparseMultivariateTaylorSeries
SparseTable
SparseUnivariateLaurentSeries
SparseUnivariatePolynomial
SparseUnivariatePuiseuxSeries
SparseUnivariateSkewPolynomial
SparseUnivariateTaylorSeries
SplitHomogeneousDirectProduct
SplittingNode
SplittingTree
SquareFreeRegularTriangularSet
SquareMatrix
Stack
Stream
String
StringTable
SubSpace
SubSpaceComponentProperty
SuchThat
Switch
Symbol
SymbolTable
SymmetricPolynomial
Table
Tableau
TaylorSeries
TexFormat
TextFile
TheSymbolTable
ThreeDimensionalMatrix
ThreeDimensionalViewport
ThreeSpace
Timer
Tree
TubePlot
Tuple
TwoDimensionalArray
TwoDimensionalViewport
Union
UnivariateLaurentSeries
UnivariateLaurentSeriesConstructor
UnivariatePolynomial
UnivariatePuiseuxSeries
UnivariatePuiseuxSeriesConstructor
UnivariatePuiseuxSeriesWithExponentialSingularity
UnivariateSkewPolynomial
UnivariateTaylorSeries
UniversalSegment
Variable
Vector
Void
WeightedPolynomials
WuWenTsunTriangularSet
XDistributedPolynomial
XPBWPolynomial
XPolynomial
XPolynomialRing
XRecursivePolynomial
AlgebraicFunction
AlgebraicHermiteIntegration
AlgebraicIntegrate
AlgebraicIntegration
AlgebraicManipulations
AlgebraicMultFact
AlgebraPackage
AlgFactor
AnnaNumericalIntegrationPackage
AnnaNumericalOptimizationPackage
AnnaOrdinaryDifferentialEquationPackage
AnnaPartialDifferentialEquationPackage
AnyFunctions1
ApplyRules
ApplyUnivariateSkewPolynomial
AssociatedEquations
AttachPredicates
BalancedFactorisation
BasicOperatorFunctions1
BezoutMatrix
BoundIntegerRoots
BrillhartTests
CartesianTensorFunctions2
ChangeOfVariable
CharacteristicPolynomialInMonogenousalgebra
CharacteristicPolynomialPackage
ChineseRemainderToolsForIntegralBases
CoerceVectorMatrixPackage
CombinatorialFunction
CommonDenominator
CommonOperators
CommutUnivariatePolynomialCategory
ComplexFactorization
ComplexFunctions2
ComplexIntegerSolveLinearPolynomialEquation
ComplexPattern
ComplexPatternMatch
ComplexRootFindingPackage
ComplexRootPackage
ComplexTrigonometricManipulations
ConstantLODE
CoordinateSystems
CRApackage
CycleIndicators
CyclicStreamTools
CyclotomicPolynomialPackage
d01AgentsPackage
d01WeightsPackage
d02AgentsPackage
d03AgentsPackage
DefiniteIntegrationTools
DegreeReductionPackage
DiophantineSolutionPackage
DirectProductFunctions2
DiscreteLogarithmPackage
DisplayPackage
DistinctDegreeFactorize
DoubleFloatSpecialFunctions
DoubleResultantPackage
DrawComplex
DrawNumericHack
DrawOptionFunctions0
DrawOptionFunctions1
e04AgentsPackage
EigenPackage
ElementaryFunction
ElementaryFunctionDefiniteIntegration
ElementaryFunctionLODESolver
ElementaryFunctionODESolver
ElementaryFunctionSign
ElementaryFunctionStructurePackage
ElementaryIntegration
ElementaryRischDE
ElementaryRischDESolver
EllipticFunctionsUnivariateTaylorSeries
EquationFunctions2
ErrorFunctions
EuclideanGroebnerBasisPackage
EvaluateCycleIndicators
ExpertSystemContinuityPackage
ExpertSystemContinuityPackage1
ExpertSystemToolsPackage
ExpertSystemToolsPackage1
ExpertSystemToolsPackage2
ExpressionFunctions2
ExpressionSpaceFunctions1
ExpressionSpaceFunctions2
ExpressionSpaceODESolver
ExpressionToOpenMath
ExpressionToUnivariatePowerSeries
ExpressionTubePlot
FactoredFunctions
FactoredFunctions2
FactoredFunctionUtilities
FactoringUtilities
FGLMFCanPackage
FindOrderFinite
FiniteDivisorFunctions2
FiniteFieldFunctions
FiniteFieldHomomorphisms
FiniteFieldPolynomialPackage
FiniteFieldPolynomialPackage2
FiniteFieldSolveLinearPolynomialEquation
FiniteLinearAggregateFunctions2
FiniteLinearAggregateSort
FiniteSetAggregateFunctions2
FloatingComplexPackage
InnerModularGcd
InnerMultFact
InnerNormalBasisFieldFunctions
InnerNumericEigenPackage
InnerNumericFloatSolvePackage
InnerPolySign
InnerPolySum
InnerTrigonometricManipulations
InputFormFunctions1
IntegerBits
IntegerCombinatoricFunctions
IntegerFactorizationPackage
IntegerLinearDependence
IntegerNumberTheoryFunctions
IntegerPrimesPackage
IntegerRetractions
IntegerRoots
IntegerSolveLinearPolynomialEquation
IntegralBasisPolynomialTools
IntegralBasisTools
IntegrationResultFunctions2
IntegrationResultRFToFunction
IntegrationResultToFunction
IntegrationTools
InternalPrintPackage
InternalRationalUnivariateRepresentationPackage
InverseLaplaceTransform
IrredPolyOverFiniteField
IrrepsSymNatPackage
KernelFunctions2
Kovacic
LaplaceTransform
LazardSetSolvingPackage
LeadingCoeffDetermination
LexTriangularPackage
LinearDependence
LinearOrdinaryDifferentialOperatorFactorizer
LinearOrdinaryDifferentialOperatorsOps
LinearPolynomialEquationByFractions
LinearSystemMatrixPackage
LinearSystemMatrixPackage1
LinearSystemPolynomialPackage
LinGroebnerPackage
LiouvillianFunction
ListFunctions2
ListFunctions3
ListToMap
MakeBinaryCompiledFunction
MakeFloatCompiledFunction
MakeFunction
MakeRecord
MakeUnaryCompiledFunction
MappingPackage1
MappingPackage2
MappingPackage3
MappingPackageInternalHacks1
MappingPackageInternalHacks2
MappingPackageInternalHacks3
MatrixCategoryFunctions2
MatrixCommonDenominator
MatrixLinearAlgebraFunctions
MergeThing
MeshCreationRoutinesForThreeDimensions
ModularDistinctDegreeFactorizer
ModularHermitianRowReduction
MonoidRingFunctions2
MonomialExtensionTools
MoreSystemCommands
MPolyCatFunctions2
MPolyCatFunctions3
MPolyCatPolyFactorizer
MPolyCatRationalFunctionFactorizer
MRationalFactorize
MultiFiniteFactorize
MultipleMap
MultiVariableCalculusFunctions
MultivariateFactorize
MultivariateLifting
MultivariateSquareFree
NagEigenPackage
NagFittingPackage
NagIntegrationPackage
NagInterpolationPackage
NagLapack
NagLinearEquationSolvingPackage
NAGLinkSupportPackage
NagMatrixOperationsPackage
NagOptimisationPackage
NagOrdinaryDifferentialEquationsPackage
NagPartialDifferentialEquationsPackage
NagPolynomialRootsPackage
NagRootFindingPackage
NagSeriesSummationPackage
NagSpecialFunctionsPackage
NewSparseUnivariatePolynomialFunctions2
NonCommutativeOperatorDivision
NoneFunctions1
NonLinearFirstOrderODESolver
NonLinearSolvePackage
NormalizationPackage
NormInMonogenicAlgebra
NormRetractPackage
NPCoef
NumberFieldIntegralBasis
NumberFormats
NumberTheoreticPolynomialFunctions
Numeric
NumericalOrdinaryDifferentialEquations
NumericalQuadrature
NumericComplexEigenPackage
NumericContinuedFraction
NumericRealEigenPackage
NumericTubePlot
OctonionCategoryFunctions2
ODEIntegration
ODETools
OneDimensionalArrayFunctions2
OnePointCompletionFunctions2
OpenMathPackage
OpenMathServerPackage
OperationsQuery
OrderedCompletionFunctions2
OrderingFunctions
OrthogonalPolynomialFunctions
OutputPackage
PadeApproximantPackage
PadeApproximants
PAdicWildFunctionFieldIntegralBasis
ParadoxicalCombinatorsForStreams
ParametricLinearEquations
ParametricPlaneCurveFunctions2
ParametricSpaceCurveFunctions2
ParametricSurfaceFunctions2
PartialFractionPackage
PartitionsAndPermutations
PatternFunctions1
PatternFunctions2
PatternMatch
PatternMatchAssertions
PatternMatchFunctionSpace
PatternMatchIntegerNumberSystem
PatternMatchIntegration
PatternMatchKernel
PatternMatchListAggregate
PatternMatchPolynomialCategory
PatternMatchPushDown
PatternMatchQuotientFieldCategory
PatternMatchResultFunctions2
PatternMatchSymbol
PatternMatchTools
Permanent
PermutationGroupExamples
PiCoercions
PlotFunctions1
PlotTools
PointFunctions2
PointPackage
PointsOfFiniteOrder
PointsOfFiniteOrderRational
PointsOfFiniteOrderTools
PolToPol
PolyGroebner
PolynomialAN2Expression
PolynomialCategoryLifting
PolynomialCategoryQuotientFunctions
PolynomialComposition
PolynomialDecomposition
PolynomialFactorizationByRecursion
PolynomialFactorizationByRecursionUnivariate
PolynomialFunctions2
PolynomialGcdPackage
PolynomialInterpolation
PolynomialInterpolationAlgorithms
PolynomialNumberTheoryFunctions
PolynomialRoots
PolynomialSetUtilitiesPackage
PolynomialSolveByFormulas
PolynomialSquareFree
PolynomialToUnivariatePolynomial
PowerSeriesLimitPackage
PrecomputedAssociatedEquations
PrimitiveArrayFunctions2
PrimitiveElement
PrimitiveRatDE
PrimitiveRatRicDE
PrintPackage
PseudoLinearNormalForm
PseudoRemainderSequence
PureAlgebraicIntegration
PureAlgebraicLODE
PushVariables
QuasiAlgebraicSet2
QuasiComponentPackage
QuaternionCategoryFunctions2
QuotientFieldCategoryFunctions2
RadicalEigenPackage
RadicalSolvePackage
RadixUtilities
RandomDistributions
RandomFloatDistributions
RandomIntegerDistributions
RandomNumberSource
RationalFactorize
RationalFunction
RationalFunctionDefiniteIntegration
RationalFunctionFactor
RationalFunctionFactorizer
RationalFunctionIntegration
RationalFunctionLimitPackage
RationalFunctionSign
RationalFunctionSum
RationalIntegration
RationalLODE
RationalRetractions
RationalRicDE
RationalUnivariateRepresentationPackage
RealPolynomialUtilitiesPackage
RealSolvePackage
RealZeroPackage
RealZeroPackageQ
RectangularMatrixCategoryFunctions2
ReducedDivisor
ReduceLODE
ReductionOfOrder
RegularSetDecompositionPackage
RegularTriangularSetGcdPackage
RepeatedDoubling
RepeatedSquaring
RepresentationPackage1
RepresentationPackage2
ResolveLatticeCompletion
RetractSolvePackage
SAERationalFunctionAlgFactor
ScriptFormulaFormat1
SegmentBindingFunctions2
SegmentFunctions2
SimpleAlgebraicExtensionAlgFactor
SimplifyAlgebraicNumberConvertPackage
SmithNormalForm
SortedCache
SortPackage
SparseUnivariatePolynomialFunctions2
SpecialOutputPackage
SquareFreeQuasiComponentPackage
SquareFreeRegularSetDecompositionPackage
SquareFreeRegularTriangularSetGcdPackage
StorageEfficientMatrixOperations
StreamFunctions1
StreamFunctions2
StreamFunctions3
StreamInfiniteProduct
StreamTaylorSeriesOperations
StreamTranscendentalFunctions
StreamTranscendentalFunctionsNonCommutative
StructuralConstantsPackage
SturmHabichtPackage
SubResultantPackage
SupFractionFactorizer
SymmetricFunctions
SymmetricGroupCombinatoricFunctions
SystemODESolver
SystemSolvePackage
TableauxBumpers
TabulatedComputationPackage
TangentExpansions
TemplateUtilities
TexFormat1
ToolsForSign
TopLevelDrawFunctions
TopLevelDrawFunctionsForAlgebraicCurves
TopLevelDrawFunctionsForCompiledFunctions
TopLevelDrawFunctionsForPoints
TopLevelThreeSpace
TranscendentalHermiteIntegration
TranscendentalIntegration
TranscendentalManipulations
TranscendentalRischDE
TranscendentalRischDESystem
TransSolvePackage
TransSolvePackageService
TriangularMatrixOperations
TrigonometricManipulations
TubePlotTools
TwoDimensionalPlotClipping
TwoFactorize
UnivariateFactorize
UnivariateLaurentSeriesFunctions2
UnivariatePolynomialCategoryFunctions2
UnivariatePolynomialCommonDenominator
UnivariatePolynomialDecompositionPackage
UnivariatePolynomialDivisionPackage
UnivariatePolynomialFunctions2
UnivariatePolynomialMultiplicationPackage
UnivariatePolynomialSquareFree
UnivariatePuiseuxSeriesFunctions2
UnivariateSkewPolynomialCategoryOps
UnivariateTaylorSeriesFunctions2
UnivariateTaylorSeriesODESolver
UniversalSegmentFunctions2
UserDefinedPartialOrdering
UserDefinedVariableOrdering
UTSodetoools
VectorFunctions2
ViewDefaultsPackage
ViewportPackage
WeierstrassPreparation
WildFunctionFieldIntegralBasis
XExponentialPackage
ZeroDimensionalSolvePackage
Chapter 11

Research Topics

These are included here as ideas that may get expanded in more detail later.

11.1 Proofs

The goal would be to prove that Axiom’s algorithms are correct.
For instance, show that the GCD algorithm is correct. This involves several levels of proof.
At one level we need to prove that the GCD algorithm is mathematically correct and that
it terminates. This can be picked up from the literature.
A second level of correctness involves proving that the implementation of the algorithm is
correct. This involves using something like ACL2 [KMJ00] and proof of the common lisp
implementation.
A third level is to show that the binary implementation conforms to the semantics of the
common lisp implementation. This involves using something like Function Extraction (FX)
[LMW79] to extract the machine-level behavior of the program and comparing it to the
specification.

11.2 Indefinites

There are times when it would be convenient to write algorithms in terms of indefinite values.
For instance, we would like to be able to declare that X and Y are matrices and compute
X*Y symbolically. We would like to be able to do the same with arbitrary integers, I and
J. In general, for a given domain we would like to create domain elements that are not fully
specified but have the computation proceed with these “indefinite” values.

11.3 Provisos

We would like to create “provisos” on statements such as:
\[
\frac{1}{x} \quad \text{provided } x \neq 0
\]
We would then like to rewrite this in terms of intervals to create three “continuations” where each continuation is a separate domain of computation (and could thus be computed in parallel). So for the above example we would generate:

\[
\frac{1}{x} \text{ such that } x \in [-\infty, 0) \\
\frac{1}{x} \text{ such that } x \in (0, 0) \\
\frac{1}{x} \text{ such that } x \in (0, \infty]
\]

When a new proviso is added, for instance, when we divide by \( y \) then there would be further subdivision of the computation, forming a tree:

\[
\frac{1}{xy} \text{ such that } x \in [-\infty, 0) \text{ and } y \in [-\infty, 0) \\
\frac{1}{xy} \text{ such that } x \in (0, 0) \text{ and } y \in [-\infty, 0) \\
\frac{1}{xy} \text{ such that } x \in (0, \infty] \text{ and } y \in [-\infty, 0) \\
\frac{1}{xy} \text{ such that } x \in [-\infty, 0) \text{ and } y \in (0, 0) \\
\frac{1}{xy} \text{ such that } x \in (0, 0) \text{ and } y \in (0, 0) \\
\frac{1}{xy} \text{ such that } x \in (0, \infty] \text{ and } y \in (0, 0) \\
\frac{1}{xy} \text{ such that } x \in [-\infty, 0) \text{ and } y \in (0, \infty] \\
\frac{1}{xy} \text{ such that } x \in (0, 0) \text{ and } y \in (0, \infty] \\
\frac{1}{xy} \text{ such that } x \in (0, \infty] \text{ and } y \in (0, \infty]
\]

Interesting questions arise, such as how to recover the function over the real line. Of course, the domain and range are not restricted to the real line in general but could, for instance, range over the complex plane.

Note that the provisos need not be an interval. They could be anything such as a polynomial or a property like “\( f(x) \) is entire”.
Chapter 12

Makefile

BOOK=${SPD}/books/bookvol6.pamphlet

# this is where to put the various commands
OUT= ${MNT}/${SYS}/bin
OUTLIB= ${MNT}/${SYS}/lib

# this is where we hid the libspad library
LIB= ${OBJ}/${SYS}/lib

# this is where the documentation ends up
LDFLAGS= -L${LIB} -lspad ${LDF}

all: announce ${OUTLIB}/session ${OUTLIB}/spadclient ${OUT}/sman \
     ${OUT}/axiom finish

announce:
  echo ==============================================================
  echo Making sman, session, spadclient, axiom bookvol6
  echo ==============================================================

finish:
  echo ==============================================================
  echo Finished sman, session, spadclient, axiom bookvol6
  echo ==============================================================

${OUTLIB}/session: ${LIB}/libspad.a
  echo 1 making ${OUTLIB}/session from ${BOOK}
  (cd ${LIB} ; \$
    ${BOOKS}/tanglec ${BOOK} session.c >session.c ; \$
    ${CC} ${CCF} -o ${OUTLIB}/session session.c libspad.a )

${OUTLIB}/spadclient: ${LIB}/libspad.a
  echo 2 making ${OUTLIB}/spadclient from ${BOOK}
  (cd ${LIB} ; \$

\$\{BOOKS\}/tanglec \$\{BOOK\} spadclient.c >spadclient.c ; \\
\$\{CC\} \$\{CCF\} -o \$\{OUTLIB\}/spadclient spadclient.c libspad.a \\
\n\$\{OUT\}/axiom: \$\{BOOK\}
@ echo 3 making \$\{OUT\}/axiom from \$\{BOOK\}
@ (cd \$\{OUT\} ; \$\{BOOKS\}/tanglec \$\{BOOK\} axiomcmd >axiom )
@ chmod +x \$\{OUT\}/axiom

\$\{OUT\}/sman: \$\{LIB\}/libspad.a
@ echo 4 making \$\{OUT\}/sman from \$\{BOOK\}
@ (cd \$\{LIB\} ; \\
   \$\{BOOKS\}/tanglec \$\{BOOK\} sman.c >sman.c ; \\
   \$\{CC\} \$\{CCF\} -o \$\{OUT\}/sman sman.c libspad.a )

----------
Bibliography


