KOZEL: Kernel Organization Zappy Environment for Linux

Konstantin Beznosov
Sergey Fedorishin
School of Computer Science
Florida International University
beznosov@cs.fiu.edu
sfedor01@cs.fiu.edu

April 28, 1997

Abstract

This report describes application domain, design and usage of Kernel Organization Zappy Environment for Linux (KOZEL, pronounced "kozz'jol") developed during a term project for "Expert Systems" cource #CEN5120 tought by Dr. Pelin in Spring of 1997 at School of Computer Science[1], Florida International University[2]. We present problem the system is designed to solve, discuss a conceptual view of the system architecture, give a detailed picture of its implementation and describe usage of the system.

The document is available in electronic form at http://www.cs.fiu.edu/~beznosov/doc/kozel.

[1] http://hpdrc.cs.fiu.edu
[2] http://www.fiu.edu

Contents

1 Introduction

Today task of configuring Operating System kernel requires not only knowledge of installed on the system hardware but also awareness of all those numerous parameters that inter-depend on each other. It is especially true with Linux[3] kernel that uses the same source tree to be built on 6 different architectures as well as notebooks and laptops[4]. It is important as well to find an optimal configuration, where the kernel will contain only those parts that are really needed.

Kozel is an expert tool that allows to automate a process of Linux kernel configuration (during a building phase) by asking simple questions to a user and trying to find optimal configuration. The program can be distributed under conditions of GNU General Public License version 2 re-printed in on page 27 of this report.

The remainder of this report is organized as follows. Section 2 describes the application domain of the system. Section 3 discuses available for today solutions. Section 4 presents the proposed solution. System design is explored in Section 5. Sample session with the system is illustrated in Section 7. Conclusions are contained in Section 6. *Kozel* source code is listed in Appendix A . Distribution license the program is registered under are available in Appendix B.

2 Application Domain

Expert tool *kozel* is developed to assist in configuring Linux[5] kernel when the kernel is built from source code. More than 300 parameters are involved in kernel configuration. They define various features of the kernel from such global characteristics as whether to build TCP/IP stack into the kernel to such miscellaneous once as Direct Memory Access channel number for a sound card.

There are dependencies among parameters. It allows for example to configure sound card DMA channel number only if a sound card is configured to be supported by the particular kernel configuration. Total parameter space is a directed graph, which is not connected.

Most of parameters define whether particular parts responsible for specific services or functionality will be built into the kernel. Those parameters can have either yes/no; yes as a module [m], or yes as a monolithic part [y], or no [n]. There are also parameters that can take integer or hexademical values. Those values are closely related to the particular hardware configuration of computer system .

3 Available Solution

Commonly accepted approach to configuration phase is a set of recursive scripts that ask a kernel configurator (further in the report *user*) to define configuration parameters or provide their values id applicable. Those scripts go from *"root"* parameters (those that do not depend on anything else) to "leaf" once (those who are not dependant on). For example, if a user answered "no" to CONFIG_ISDN the user will not be asked about any ISDN device.

[3] http://sunsite.unc.edu/linux

[4] http://www.hal-pc.org/ davidl/linux/linux.config.html

[5] http://sunsite.unc.edu/linux

Such an approach has its advantages. It allows to behave pseudo intelligently and to eliminate many “dead routes”. It’s also simple to implement. As a matter of fact, this approach is implemented in Unix shell.

It also has tradeoffs. First is that a user has to know the whole hierarchy of parameters and define correctly those near to the root to be asked later about those further from the root. For example, if the system has PCI network card, the user has to answer “yes” somewhere at the beginning of the configuration session to the question about PCI capabilities of the computer system in general to have a chance to be asked about his/her PCI network adapter later.

Another drawback is that configuration program “talks” to the user in terms of parameters and not in terms of actual hardware available on the system and real services the operating system is supposed to provide. It is easy for a human been to get confused and to make mistakes in this case.

4 Proposed Solution

In the proposed solution, the configuration tool interviews a user about actual hardware the computer system has or will have and actual services the operating system is supposed to provide. In other words, a configuration comes from “leaf” parameters to “root” parameters. The user is asked what hardware is available on the system and what services the system will provide, and the rest is configured automatically including definition of “root” parameters that has to be defined for the system to provide required services.

4.1 Session Phases

A session of configuring a kernel consists of generating and loading dependencies into the knowledge base, interviewing a user, configuring hierarchy of parameters, and generating the result of the session. Bellow is a description of each step.

4.1.1 Loading Of Dependencies

Dependencies among paramaters are generated dynamically each time before interview with a user begins. They are extracted from the configuration scripts provided with the kernel source code.

4.1.2 Interviewing The User

Interview with the user begins when the user choses command *configure* from the main menu. During the interview the user is asked about hardware components of the system, their types and, when it is nessessary, about specific for particular hardware parameters. Example of such interview is provided in Section 7 on page 6.

4.1.3 Configurating The System

After the interview is over, the program attempts to resolve dependencies for thoses parameters that were instantiated during the previous phase. Conflicts are detected and resolved at this step as well.

4.1.4 Creating Configuration File

After all dependencies found and conflicts are resolved, the result of the session – configuration file in format acceptable by native Linux kernel building tools is generated. Bellow is an example of such a file:

```
CONFIG_EXPERIMENTAL=y
CONFIG_CD_NO_IDESCSI=y
CONFIG_NET_ISA=y
CONFIG_NET_ETHERNET=y
CONFIG_NETDEVICES=y
CONFIG_NET=y
CONFIG_MOUSE=y
CONFIG_BLK_DEV_IDE=y
CONFIG_PRINTER=y
CONFIG_PSMOUSE=y
CONFIG_SERIAL=y
CONFIG_INET=y
CONFIG_PPP=y
CONFIG_DEPCA=y
CONFIG_SBPCD=y
CONFIG_BLK_DEV_FD=y
CONFIG_BINFMT_JAVA=y
CONFIG_KERNEL_ELF=y
CONFIG_M586=y
```

5 System Design

Tool *Kozel* is designed as follows:

1. False driven loop *mainloop* does main menu. It prints list of possibilities and calls *get_answer/1* and *do_command/1* to recognize valid response and perform an appropriate command.
2. Service features like print list of anything, get response, make a pause are provided by *printlist/1*, *means/2*, *pause*, *get_single_char/1* and others.
3. User can ask for help and *do_help* will shed some light.
4. The system can operate in 3 modes: manual consulting, automatic consulting and conflicts resolving. Default mode is automatic and only this mode is used in a production system. Manual mode is used exclusively for debugging and test phases.

The predicates involved in dialog are *config_not_auto*, *config_linux/1*,

config/1, *intro/1* and *define/1*. The predicate *define/1* starts definition process.

5.1 Knowledge Base

The knowledge base consists of list of dependencies among paramataers and their definitions, as well as groupping of some parameters according to the type of system hardware they define. Parameters and dependencies among them are generated automatically before each interview with the user. Groups of variables, on the other hand, are a part of knowledge base that has to by created by aquiring knowledge from a human expert.

5.1.1 Automatically Generated

List of paramaters and dependencies among them is generated by a program *config2rules* (see page 22 for the program listing) written in Perl that parses configuration scripts of Linux kernel source code. The output of is loaded as a set of facts into *kozel* databases. Belllow is a fragment of such a database:

```
variable('IPX_INTERN', b, 'Full internal IPX network', n, undefined ).

if 'IPX_INTERN' is_wanted and 'IPX' is_not n then 'IPX_INTERN'.
```

It means that parameter `IPX_INTERN` has type boolean, help message `'Full internal IPX network'` , and its default value is set to *n* (means no), as well as current value is `undefined`. The second line means that to set this paramater to "yes" it has to be expliciltly set during an interview with the user or some other parameter (which depends on `IPX_INTERN`) should be set to "yes". Also if `IPX_INTERN` is set another parameter `IPX` should be set too.

5.1.2 Acquired From A Human Expert.

Most of kernel parameters that are mapped into system hardware componennts or services, has to be grouped into meaningfull sets to facilitate interview with the user. The step of groupping parametrs is done manually via knowledge aqusion from a human expert. Current version of systen kozel has the following groups of parameters:

1. Hard disk types.
2. CD-ROM types.
3. Network interfaces.
4. Network connections.
5. Modem types.
6. Printer types.
7. Mouse types.
8. Tape device types.

5.2 Interview With End User

During the interview with the end user, the user is asked to choose what particular types of hardware the computer has and what types of services the operating system is supposed to provide.

The user is asked about listed above groups via predicate *ask_type_of_hardware/2*. Questions during the interview are asked via predicate *askTermD/4*.

5.3 Kernel Configuration

Manual consulting assumes that knowledge is organized in the hierarchy when kernel parameters or other atomic facts are clustered into groups and user scans hierarchy tree in a breadth search manner. So it is backward chaining meaning we start from goal “build valid configuration” and go deeper and deeper towards facts.

For each atomic parametersystem performs check if some conflicts exist in a forward chaining manner. Also, which is more valuable from the application domain point of view it sets other variables to default values whenever it found dependency from an undefined so far variable.

Predicate *resolve* starts post-configuration to find conflicts and set other variables starting from those already defined by user. It uses *resolve_var/4* to do that. Predicate *fflag/2* maintains operation mode. Automatic consulting does the same but without prompts. It uses *config_auto* as a starting point.

The consultation itself is carried out by *definevar*/1, process_var/1, *process_condition_of/2*, *setvar/3*, *set_variable_value/2* and *evaluate/3*.

5.4 Output Of Session Results

All parameters in the database are checked when session results are generated. If a parameter is set to some of possitive values it is written into the generated output. An example of such output is shown on page **??**.

6 Conclusions

Expert tool *Kozel* presents an alternative solution to a teduous and errorprone task to configure kernel of Linux operating system in right and optimal way.

7 Example of Session With The User

A KOZEL Source Code

A.1 Kozel.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Kernel Organization Zappy Environment for Linux            %
%                                           %  %                         %
%              %    %   %%%%%%%   %%%%%%   %%%%%%    %%%%%%              %
%              %   %    %     %        %   %        %     %              %
%              %  %     %     %        %   %        %     %              %
%              % %      %     %       %    %        %     %              %
%              %%       %     %      %     %%%%%    %     %              %
%              % %      %     %       %    %        %     %              %
%              %  %     %     %        %   %        %     %              %
%              %   %    %     %        %   %        %     %              %
%              %    %   %%%%%%%   %%%%%%   %%%%%%   %     %              %
%                                                                        %
%                                                                        %
%                                                                        %
%                                                                        %
%                                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FILENAME
%$Id: //depot/home/beznosov/Classes/CEN5120/project/linux-kernel/kozel/kozel.pl#8$
%
% DESCRIPTION
% Configures Linux (v2.x) kernel intellegently.
%
% AUTHOR
%       Konstantin Beznosov (http://www.cs.fiu.edu/~beznosov)
% Sergey Fedorisihin (http://www.cs.fiu.edu/~sfedor01)
%
% CREATION DATE
%       4/2/97
%
% COPYRIGHT
%       1997 (c) Konstantin Beznosov and Sergey Fedorishin
%                         All rights Reserved
%
% WARRANTY
%  THIS SOFTWARE IS PROVIDED BY KONSTANTIN BEZNOSOV AND SERGEY FEDORISHIN
%  ‘‘AS IS’’ AND ANY  EXPRESSED 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 KONSTANTIN
%  BEZNOSOV AND SERGEY FEDORISHIN OR THEIR 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.
%
%
% LICENSE
%       GNU General Public License v. 2
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---------------------------------------------------------------
% Operators
:- op(990,fx,if).
:- op(985,xf,is_wanted).
:- op(980,xfx,then).
:- op(970,xfx,or).
:- op(960,xfy,and).
```

```
:- op(950,fx,not).
:- op(690,xfy,is).
:- op(950,xfy,is_not).
:- op(720,xfx,set_to).

%-----------------------------------------------------------------

% gop: just do it! (russian slang)
gop :- mainloop.

% mainloop: main false driven loop, calls a_cycle
mainloop :- repeat, a_cycle.

% a_cycle: command cycle, calls printlist, get_answer, do_command
a_cycle :-
  nl, write( 'XpertS: course project'), nl,
  printlist( ['Option List (please take one):'
            , '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'
%           , '(angle brackets <> mean command argument)'
            , ''
            , 'quit              -- Exit from the program'
            , 'help              -- Find helpful information on the program'
            , 'config            -- Start configuration of the kernel'
%           , 'autoconfig         -- automatically define rest of variables'
%           , 'undefine           -- undefine variable or section'
%           , 'load (file)        -- load knowledge base'
%           , 'save (file)        -- save knowledge base'
%           , '!command           -- run operating system command'
            , 'resolve           -- resolve dependecnies'
            , 'shell             -- Execute shell command'
             ]
  ), nl,
  write('Your turn: '),
  get_answer(Answer),!,
  do_command(Answer),
  !, fail.

% help facility
do_help :-
  printlist( ['This system is to configure Linux ...'
            , 'and it was developed by ...'
            , 'under provisions of ...'
            , 'and by the way, the Linux is ...'
            , 'and configuration of Linux consists of:'
            , 'a) '
            , 'b)'
            , 'c)'
            , 'd)'
            , '...'
            , 'and more help available upon request: give us a call ...'
             ]
  ), pause.

% main goal
config_not_auto :-  fflag( auto, yes)
                  , retract( fflag( auto, _))
                  , fail
                ;   asserta( fflag( auto, no))
                  , config_linux.

% we just set flag and go
config_auto :-  fflag( auto, no)
              , retract( fflag( auto, _))
              , fail
            ;   asserta( fflag( auto, yes))
              , config_linux.
```

```
config_linux :- load_kb,
                ask_user,
                resolve,
                write_result,
                pause.


old_config_linux:-
                 load_kb
               , config( general)
               , config( net)
               , config( protocols)
               , config( drivers)
               , config( misc)
.

% config procedure
% by typing anything but yes/y in response to intro
%   user can skip some thing and go directly to another
config( Something) :- intro( Something)
                    , define( Something).

config( _).

% ask user if he wants to continue with this section/variable
% or proceed if autopilot on duty
% no default behaviour for auto flag assumed
intro( _ )        :- fflag( auto, yes).
intro( Something) :- fflag( auto, no),
                     printlist([ 'You are about to define'
                       , Something
                       , 'do you want to continue?'
                       ]),
                     get_answer( Yesno), !,
                     Yesno == yes.

% load_kb/0 Loads knowledge base
load_kb:-
     ['linux_kernel_config.pl']
    , true.

%------------------------ source database -------------------------

fflag( auto, yes).
fflag( modules, yes).

% for your convenience service predicates
default_value( Var, X) :- variable( Var, _, _, X, _).
undefined( Var)        :- variable( Var, _, _, _, undefined).
hasvalue( Var, Value)  :- variable( Var, _, _, _, Value)
                        , Value \== undefined.

%-------------------- basic definition technique --------------------

% for process_condition_of/2 'if' clause defines branch
% case 1: unconditional setting means no dependencies: nothing to do
process_condition_of( V, Ans) :-
          if V is_wanted then V
        , Ans is 1.

% case 2: go and find out if precondition(s) satisfy or set it (them) settled
process_condition_of( V, Ans ) :-
          if V is_wanted and Cond then V
        , evaluate( Cond, y, Ans)
.
```

```
% case 3
process_condition_of( V, Ans ) :-
          if V is_wanted and Cond1 or Cond2 then V
        , evaluate( Cond1 or Cond2, y, Ans)
.

process_condition_of( _, Ans ) :- Ans is 0.
%      write('by default Ans is 0 in process_condition_of'), nl.


% resolve: scans database and finds errors and contradictions
%          for defined variables, it doesn't touch undefined or set to n
% assumes no more than one instance for each variable
% if you want it to stop after 1st bad case uncomment cut
resolve :- variable( Var, Vartype, _, Default, Value)
         , Value \== undefined
         , Value \== n
%         , !
         , resolve_var( Var, Vartype, Default, Value)
.
resolve.

% resolve_var: checks if given variable ok, ie can be set by auto_setvar
%resolve_var( Var, Vartype, Default, Value) :-
resolve_var( Var, _,_,_) :-
                           process_condition_of( Var, Ans)
                         , !, Ans == 1
%                         , auto_setvar( Var, Vartype, Default, Value)
                         , !, fail
.

% auto_setvar: determines value to be assigned by default
% yes or m for type t, yes for type b, default value for others (hex and int)
auto_setvar( Var, t, _, Val) :- fflag( modules, no), Val \== y,
                                setvar( Var, y, 1).
auto_setvar( Var, t, _, Val) :- fflag( modules, yes), Val \== m,
                                setvar( Var, m, 1).
auto_setvar( Var, b, _, Val) :- Val \== y, setvar( Var, y, 1).
auto_setvar( Var, _, Def, Val) :- Val \== Def, setvar( Var, Def, 1).


% definevar: defines variables via dialog mainly

% case 1: if unknown, define it
% user is asked if he wants to define this variable,
%   later only valid answers will be accepted in infinite loop
definevar( V ) :- fflag( auto, no)
                , undefined( V)
                , intro( V)
                , process_var(V)
.

% case 2: automatic mode and undefined var: skip it
definevar( V ) :- fflag( auto, yes)
                , undefined( V)
.

% case 3: automatic mode and defined var: go find others
definevar( V ) :- fflag( auto, yes)
                , hasvalue( V, Value)
                , auto_process_var( V, Value)
.

% same as in config: user can skip some variables or
%                   system will skip already defined
```

```
% case 4: manual mode, undefined var: skip it and go for others
definevar( _).
% :- write('default for definevar '), write(V), nl.


% process_var: we consider all variables askable
% get_reply keeps asking until valid reply provided
% but process_condition_of may fail and variable remains undefined
process_var(V) :- variable( V, Vtype, Prompt,_,_)
                , express_domain( Vtype, Dom)
                , printlist( [ 'Please make your choice for'
                             , V
                             , 'which is'
                             , Prompt
                             , 'possible values are'
                             , Dom
                             ])
                , get_reply( Ans, Vtype)
                , process_condition_of( V, Ok )
                , setvar( V, Ans, Ok)
.
process_var( _) :- write('default for process_var'), nl.

% auto mode for bulk processing
auto_process_var( Var, Value) :- hasvalue( Var, Oldvalue),
                  printlist( [ 'This variable'
                             , Var
                             , 'has value'
                             , Oldvalue
                             , 'while should have'
                             , Value
                             , 'please advise... '
                             ] ).

% process if evaluation suceeded
auto_process_var( Var, _) :-
                  process_condition_of( Var, Ok)
                , !
                , Ok == 1
                , variable( Var, Vartype, _, Default, _)
                , auto_setvar( Var, Vartype, Default)
.

%------------------- processing conditions --------------------

% store variable as defined with value
setvar( V, Ans, Guard) :- Guard == 1, set_variable_value( V, Ans).

setvar( V, Ans, _) :- nl, write( 'Attn! variable '),
                      write( V), write(' was not set to '), write( Ans), nl.

% we keep them in database
set_variable_value( Name, Value):-
                    retract(variable(Name, Type, Prompt, Default, _)),
                    asserta(variable(Name, Type, Prompt, Default, Value)).

% evaluate/3
% assume hasvalue/2 is opposite to undefined/1

% case 1: one variable and one value: is value OK?
evaluate( Var set_to Value, _ , Ans) :- hasvalue( Var, Value)
                                      , Ans is 1.

% case 2: variable is undefined, manual mode
evaluate( Var set_to Value, _, Ans ) :- undefined( Var)
                , fflag( auto, no)
```

```
                , printlist1( [ Var
                            , 'should be defined as'
                            , Value
                            %, 'press letter y and dot '
                            , 'To confirm, please'
                            ])
                , pause
                , process_condition_of( Var, Ans)
                , setvar( Var, Value, Ans)
.

% case 3: variable is undefined, auto mode
evaluate( Var set_to Value, _, Ans ) :- undefined( Var)
                , fflag( auto, yes)
                , process_condition_of( Var, Ans)
                , setvar( Var, Value, Ans)
.

% case 4: if value is invalid in other clauses, then complain
evaluate( Var set_to Value, y, Ans ) :-
                  printlist1( [ Var
                            , 'must be defined as'
                            , Value
                            , 'under given condition(s).'
                            , 'Please advise... '
                            ])
                , Ans is 0
                , pause
.

% case 'not': var has value, either good or bad
evaluate( Var is_not Value, _, Ans ) :- hasvalue( Var, Other),
                                    (  Other \== Value
%                                   ;  evaluate( Var set_to Value, Mode, Ans )
                                   ;    Ans is 0
                                    ).

% case 'not': when undefined, just set it
evaluate( Var is_not Value, Mode, Ans ) :-
    variable( Var, Typ, _, Def, undefined),
    process_condition_of( Var, Ans),
    !, Ans == 1,
    ( Typ == b, Value \== y,      evaluate( Var set_to y, Mode, Ans)
    ; Typ == t, Value \== y, fflag( modules, no),
                                  evaluate( Var set_to y, Mode, Ans)
    ; Typ == t, Value \== m, fflag( modules, yes),
                                  evaluate( Var set_to m, Mode, Ans)
    ; Typ == int, Value \== Def,  evaluate( Var set_to Def, Mode, Ans)
    ; Typ == hex, Value \== Def,  evaluate( Var set_to Def, Mode, Ans)
    ; ask_hard_case( Var, Typ, Def, Value, Ans)
    ).

% case 'and'
% 'and' can be evaluated recursively
evaluate( Cond1 and Cond2, X, Ans) :- evaluate( Cond1, X, Ans)
                                    , !, Ans ==1
                                    , evaluate( Cond2, X, Ans).

% case 'or'
% 'or' has different complain policy from 'and' and
%  it is also different for auto and manual modes
evaluate( Cond1 or Cond2, _, Ans) :- evaluate( Cond1, nocomplain, Ans)
        ;                            evaluate( Cond2, nocomplain, Ans)
        ;         fflag( auto, no)
                , printlist1( [ Cond1
                            , 'or'
```

```
                            , Cond2
                            , 'must be defined to be true.'
                            ])
                , pause
                , Ans is 0
.

% weird case: ask operator
ask_hard_case( Var, Typ, Def, Value, Ans) :- write('hard_case '),
                    write( Var), write( Typ),
                    write( Def), write( Value),
                    Ans is 0.


%---------------------- dialog support ----------------------
% printlist: prints list of items on the standard output
printlist(List):- clear_screen, printlist1(List).
printlist1([]).
printlist1([Item|Others]) :- nl, write(Item), printlist1(Others).

% pause: suspends and waits
pause :- nl, write('Press any key...'), ttyflush,
         get_single_char(_),
%        read(_),
         nl.

% get_answer: reads user input
get_answer( Answer) :- repeat, ttyflush, read( Something),
                       means( Something, Answer).

% abbreviations
means( yes, yes).
means( y,   yes).
means( no,  no).
means( n,   no).
means( why, why).
means( w,   why).
means( quit, quit).
means( q,    quit).
means( help  , help).
means( h     , help).
means( config, config).
means( c     , config).
means( autoconfig, autoconfig).
means( auto      , autoconfig).
means( a         , autoconfig).
means(r, resolve).
means(resolve, resolve).
means(s, shell).
means(shell, shell).
means( undefine(Var) , undef(Var)).
means( undef(Var)    , undef(Var)).
means( u(Var)        , undef(Var)).
% reject uknown answer
means( Answer, _) :- writef( '%w -- what?: ', [Answer]), fail.

% do_command: executes user command, calls load, find, oscommand
do_command(Ans) :- (
    Ans = quit             ->  halt
  ; Ans = help             ->  do_help
  ; Ans = load             ->  load_kb
%  ; Ans = save(_)       ->  not_ready  % save(File)
   ; Ans = resolve      -> resolve
  ; Ans = config           ->  config_not_auto
%  ; Ans = undef(_)       ->  not_ready  % undef(Var)
  ; Ans = shell  ->  shell_command
```

```
).

not_ready :- write('Sorry, this feature still under construction,'),
             nl, pause.


% to help user with possible values for variables
express_domain( b, 'y or n').
express_domain( t, 'y, n or m').
express_domain( int, 'integer number or d for default').

% accept reply for variable value
get_reply( Ans, Vtype):- repeat, write(' '), ttyflush, read( A),
                         rmeans( A, Vtype, Ans).

% rmeans: y, no, m, integer or hex number are valid
rmeans( y, t, y).
rmeans( n, t, n).
rmeans( m, t, m).
rmeans( y, b, y).
rmeans( n, b, n).
% explicit number or default value if d typed in
rmeans( X, int, X) :- integer(X).
rmeans( X, hex, X).
%rmeans( d, _, X) :- ??
rmeans( _, _, _) :- write( 'What?'), fail.
%-----------------------------------------------------------------------

% clears screan and puts cursor to the begining of the screen
clear_screen:-
             writef('%r', ['\n',24]),
             tty_goto(0,0).

ask_user:-
             %introduction,
                assert(fflag(giving_help, n)),
                retract_ifany(fflag(quit,y)),
                general.

retract_ifany(Clause):-
                      Clause,
                      retract(Clause),
                      fail.


retract_ifany(_).


introduction:-
             printlist([
             'You will be asked about your Linux system hardware componnets',
             'After that the program will produce configuration for your Linux kernel'
             ]),
             pause.

general:-
                      set_variable_value('KERNEL_ELF', y),!,
                      set_variable_value('BINFMT_JAVA', y),!,
             (
             processor_type,!,
             memory_size,!,
             floppy,!,
             disks,!,
             cdrom,!,
             network,!,
             modem,!,
```

```
             mouse,!,
             printer,!,
             tape,!
             )
             ;
             fflag(quit, y),
             writef('quiting...\n').

general.


%---------------------------------------------------------------------
processor_type:-      !,
              writef('What processor does your computer have ?\n'),
              askTermD('Processor Type', 386, PType_user, processor_type_help),
              determine_processor_type(PType_user, PType),
              string_concat('M', PType, Name),
              assert(variable(Name, b, 'No help', n, y)).

processor_type_help:-
                       writef('Possible choises are 386, 486, Pentium, PPro\n').


determine_processor_type('Pentium', 586).
determine_processor_type(pentium, 586).
determine_processor_type('PPro', 686).
determine_processor_type(ppro, 686).
determine_processor_type(Type,Type).

%----------------------- MEMORY SIZE ----------------------------------------
memory_size:-
               askTermD('How much main memory does your computer have (in MB) ?', 32, MemorySize),
               (
               MemorySize =< 16,
               set_variable_value('MAX_16M', y)
               ;
               set_variable_value('MAX_16M', n)
               ).

%------------------------ FLOPPY ---------------------------------------
floppy:-
                          askTermD('Does your computer have floppy ?', y, If_floppy),
                     (
                     If_floppy == y,
                     set_variable_value('BLK_DEV_FD', y)
                     ;
                     set_variable_value('BLK_DEV_FD', n)
                     ).
floppy.

%------------------------- DISKS ----------------------------------------


disks:-
                        ask_type_of_hardware(
                         disk,
                         hd_type2diskvariable
                         ).


% Hard Drives
hd_type2diskvariable('ide','BLK_DEV_IDE').
hd_type2diskvariable('pcmcia', 'BLK_DEV_IDE_PCMCIA').
hd_type2diskvariable('ali_m14xx', 'BLK_DEV_ALI14XX').
hd_type2diskvariable('dtc_2278', 'BLK_DEV_DTC2278').
hd_type2diskvariable('ht6560b', 'BLK_DEV_HT6560B').
```

```
hd_type2diskvariable('dc4030', 'BLK_DEV_PROMISE').
hd_type2diskvariable('qd6580','BLK_DEV_QD6580').
hd_type2diskvariable('umc8672','BLK_DEV_UMC8672').
hd_type2diskvariable('xt', 'BLK_DEV_XD').
hd_type2diskvariable('scsi', 'BLK_DEV_SD').

%---------------------------- CD-ROM -------------------------------
cdrom:-                 ask_type_of_hardware(
                        'CD-ROM',
                        cdrom_type2diskvariable).


% CD-ROM disks
cdrom_type2diskvariable('goldstar r420', 'GSCD').
cdrom_type2diskvariable('matsushita', 'SBPCD').
cdrom_type2diskvariable('panasonic', 'SBPCD').
cdrom_type2diskvariable('creative', 'SBPCD').
cdrom_type2diskvariable('longshine', 'SBPCD').
cdrom_type2diskvariable('teac', 'SBPCD').
cdrom_type2diskvariable('ide_atapi','BLK_DEV_IDECD').
cdrom_type2diskvariable('scsi', 'BLK_DEV_SR').

cdrom_type2diskvariable('mitsumi', CD):-
                        fflag(giving_help,n),
                        (
                        askTermD('Does it have XA/Multisession', no,
                        MultSession),
                        MultSession == no,
                        CD is 'MCD'
                        ;
                        CD is 'MCDX'
                        )
                        ;
                        true.


%---------------------------------------------------------------------
tape:-                  ask_type_of_hardware(
                        'tape drive',
                        tape_type2tapevariable).

tape_type2tapevariable(ide, 'BLK_DEV_IDE').
tape_type2tapevariable(ide_atapi, 'BLK_DEV_IDETAPE').
tape_type2tapevariable(scsi, 'CHR_DEV_ST').
tape_type2tapevariable(qic_02, 'QIC02_TAPE').
tape_type2tapevariable( ftape, 'FTAPE').

%---------------------------------------------------------------------
resolve.



network:-               Name = 'netowrka dapter',
                        ask_type_of_hardware(
                        Name,
                        net_type2netvariable),
                        network_connection(Name).


net_type2netvariable('3c501','EL1').
net_type2netvariable('3c503','EL2').
net_type2netvariable('3c505','ELPLUS').
net_type2netvariable('3c507','EL16').
net_type2netvariable('3c509','EL3').
net_type2netvariable('3c592','VORTEX').
net_type2netvariable('3c595','VORTEX').
```

```
net_type2netvariable('3c597','VORTEX').
net_type2netvariable('at1500','LANCE').
net_type2netvariable('ne2100','LANCE').
net_type2netvariable('lance','LANCE').
net_type2netvariable('pcinet32','LANCE32').
net_type2netvariable('smc','NET_VENDOR_SMC').
net_type2netvariable('wd80x3','WD80x3').
net_type2netvariable('smc_ultra','ULTRA').
net_type2netvariable('smc9194','SMC9194').
net_type2netvariable('at1700','AT1700').
net_type2netvariable('e21xx','E2100').
net_type2netvariable('depca','DEPCA').
net_type2netvariable('de10x','DEPCA').
net_type2netvariable('de200','DEPCA').
net_type2netvariable('de201','DEPCA').
net_type2netvariable('de202','DEPCA').
net_type2netvariable('de422','DEPCA').
net_type2netvariable('de203','EWRK3').
net_type2netvariable('de204','EWRK3').
net_type2netvariable('de205','EWRK3').
net_type2netvariable('etherexpress','EEXPRESS').
net_type2netvariable('etherexpresspro','EEXPRESS_PRO').
net_type2netvariable('fmv_181','FMV18X').
net_type2netvariable('fmv_182','FMV18X').
net_type2netvariable('fmv_183','FMV18X').
net_type2netvariable('fmv_184','FMV18X').
net_type2netvariable('pclan_plus27247a','HPLAN_PLUS').
net_type2netvariable('pclan_plus27247b','HPLAN_PLUS').
net_type2netvariable('pclan_27245','HPLAN').
net_type2netvariable('pclan_27xxx','HPLAN').
net_type2netvariable('hp10','HP100').
net_type2netvariable('hp100vg','HP100').
net_type2netvariable('etherteam','ETH16I').
net_type2netvariable('ne2000','NE2000').
net_type2netvariable('ne1000','NE2000').
net_type2netvariable('ni5210','NI52').
net_type2netvariable('ni6510','NI65').
net_type2netvariable('seeq8005','SEEQ8005').
net_type2netvariable('sk_g16','SK_G16').
net_type2netvariable('ansel_comm_3200','AC3200').
net_type2netvariable('apricot','APRICOT').
net_type2netvariable('de425','DE4X5').
net_type2netvariable('de434','DE4X5').
net_type2netvariable('de435','DE4X5').
net_type2netvariable('de450','DE4X5').
net_type2netvariable('de500','DE4X5').
net_type2netvariable('dc21x4x','DEC_ELCP').
net_type2netvariable('digi','DGRS').
net_type2netvariable('zenith','ZNET').
net_type2netvariable('at_lan_tec_realtek','ATP').
net_type2netvariable('de600','DE600').
net_type2netvariable('ibm_tropic','IBMTR').
net_type2netvariable('defea','DEFXX').
net_type2netvariable('defpa','DEFXX').
net_type2netvariable('arc0e','ARCNET_ETH').
net_type2netvariable('arc0s','ARCNET_1051').
net_type2netvariable('isdn','ISDN').

network_connection(Name):-
                        fflag(Name, y),
                        ask_type_of_hardware(
                        'network connection',
                        net_connection_type2netvariable).


network_connection(_).
```

```
net_connection_type2netvariable('ppp','PPP').
net_connection_type2netvariable('slip','SLIP').
net_connection_type2netvariable('plip','PLIP').
net_connection_type2netvariable('dlci','DLCI').
net_connection_type2netvariable('sdla','SDLA').
net_connection_type2netvariable('tcp_ip','INET').
net_connection_type2netvariable('radio_net','NET_RADIO').
net_connection_type2netvariable('baycom','BAYCOM').
net_connection_type2netvariable('strip','STRIP').
net_connection_type2netvariable('netrom','NETROM').
net_connection_type2netvariable('ax25','AX25').
net_connection_type2netvariable('appletalk','ATALK').
net_connection_type2netvariable('ipx','IPX_INTERN').


%---------------------------- MODEM -------------------------
modem:-
                        ask_type_of_hardware(
                        'modem',
                        serial_type2serial_variable).

serial_type2serial_variable( digiboard, 'DIGI').
serial_type2serial_variable( cyclades, 'CYCLADES').
serial_type2serial_variable( stallion, 'STALDRV').
serial_type2serial_variable( riscom8, 'RISCOM8').
serial_type2serial_variable( other, 'SERIAL').


%----------------------------- MOUSE -------------------------
mouse:-
                        ask_type_of_hardware(
                        'mouse',
                        mouse_type2mouse_variable).


mouse_type2mouse_variable( atixl, 'ATIXL_BUSMOUSE').
mouse_type2mouse_variable( logitech, 'BUSMOUSE').
mouse_type2mouse_variable( busmouse, 'BUSMOUSE').
mouse_type2mouse_variable( microsoft, 'MS_BUSMOUSE').
mouse_type2mouse_variable( ps_2, 'PSMOUSE').
mouse_type2mouse_variable( '82C710', '82C710_MOUSE').


%------------------------------ PRINTER ------------------------
printer:-
                        ask_type_of_hardware(
                        'printer',
                        printer_type2printer_variable).


printer_type2printer_variable( parallel, 'PRINTER').
printer_type2printer_variable( regular, 'PRINTER').

%-------------------------------------------------------------


%-------------------------------------------------------------------------
% ask_type_of_hardware/2 Provides uniform way to ask a user about hardware
% Installed on the system
%
ask_type_of_hardware(Name, Item):-
                        !,nl,
                        hw_type_general_help(Item, Name, 'just a list'),
                        ask_type_of_hardware(Name, Item, 0).
```

```
ask_type_of_hardware(Name, Item, Count):-
                        string_concat('What type is your computer ',Name, PromptBase0),
                        string_concat( PromptBase0, ' #', PromptBase),
                        string_concat( PromptBase, Count, Prompt),
                        askTermD(Prompt, 'nomore', Type, hw_type_help(Item, Name)),
                        aux_type_of_hardware(Name, Item, Count, Type)
                        ;
                        not fflag(quit, y).

aux_type_of_hardware(_, _, _, nomore).

aux_type_of_hardware(Name, Item, Count, Type):-
                        Type \== nomore,
                        Goal =.. [Item, Type, Var],
                        Goal,
                        (
                        set_variable_value(Var, y),
                        assert(fflag(Name, y)),
                        NCount is Count + 1,
                        ask_type_of_hardware(Name, Item, NCount)
                        )
                        ;
                        not fflag(quit, y),
                        writef('\7'),
                        writef('\nWrong type!\n'),
                        hw_type_help(Item, Name),
                        ask_type_of_hardware(Name, Item, Count).

hw_type_help(Item, Name):-
                        string_concat(Name, ' type you want info on', Prompt),
                        askTermD( Prompt, 'just a list', Type),
                        hw_type_general_help(Item, Name, Type).



hw_type_general_help(Item, Name, 'just a list'):-
                        writef('The following types of %w are available\n', [Name]),
                        retract(fflag(giving_help,_)),
                        assert(fflag(giving_help,y)),
                        Goal =.. [Item, Type, Var],
                        bagof(Type, Var ^ Goal, List),
                        retract(fflag(giving_help,_)),
                        assert(fflag(giving_help,n)),
                        write_ln(List).

hw_type_general_help(Item, Name, Type):-
                        Type \== 'just a list',
                        retract(fflag(giving_help,_)),
                        assert(fflag(giving_help,y)),
                        Goal =.. [Item, Type, Var],
                        Goal,
                        (
                        variable(Var, _, Help, _, _),
                        writef('%w : %w\n', [Type, Help]),
                        retract(fflag(giving_help,_)),
                        assert(fflag(giving_help,n))
                        )
                        ;
                        retract(fflag(giving_help,_)),
                        assert(fflag(giving_help,n)),
                        not fflag(quit, y),
                        writef('\7'),
                        writef('\nWrong type!\n'),
                        hw_type_help(Item, Name).
```

```
%-------------------------------------------------------------------------------
% Prompts user for a variable value.
% Returns typed value or default value if RETURN was pressed via Term
askTermD(Prompt, Default, Term):-
                    !,
                    askTermD(Prompt, Default, Term,0).

askTermD(Prompt, Default, Term, Help):-
                    not fflag(quit, y),
                    writef('%w [%w.](^D to abort): %f',[Prompt,Default]),
                    peek_byte(Key),
                    aux_askTermD(Key, Prompt, Default, Term, Help).


aux_askTermD(Key, Prompt, Default, Term, Help):-
                    check_user_key(Key, Prompt, Default, Term, Help)
                    ;
                    not fflag(quit, y),
                    read(Input),
                    assign_default(Input, Default, Term).



% User accepts the default value
check_user_key(10, _, Default, Term, _):-
                    skip(10),
                    write(Default),
                    assign_default(10, Default, Term).


% User typed '?' -- give him/her help
check_user_key(63, Prompt, Default, Term, 0):-
                    skip(10),
                    writef('No help available on this item\n'),
                    askTermD(Prompt, Default, Term, 0).

check_user_key(63, Prompt, Default, Term, Help):-
                    skip(10),
                    Help \== 0,
                    Help,
                    askTermD(Prompt, Default, Term, Help).

check_user_key(-1,_,_,_,_):-
                    retract_ifany(fflag(giving_help, _)),
                    assert(fflag(quit, y)),
                    writef('\n\7You pressed ^D -- aboring the interview\n'),
                    pause,
                    !,fail.

check_user_key(_,_,_,_,_):-
                    fail.

%-------------------------------------------------------------------------------
% Assigns either default value if the 1st parameter and atom end_of_file, or
% any other value given in the 1st parameter
assign_default(end_of_file, Default, Default):-
   nl,
   !.

assign_default(10, Default, Default):-
   nl,
   !.

assign_default(Value, _, Value):-
   Value \== end_of_file,
   Value \== 10.
```

```
%-------------------------------------------------------------------------------------------


write_result:-
                    askTermD('File to write configuration results to', 'config', OutFile),
                    tell(OutFile),
                    print_all_defined_vars,
                    told,
                    review_result(OutFile).

review_result(OutFile):-
                    askTermD('Do you want to review the result ?', yes, If_review),
                    (
                    If_review \== yes
                    ;
                    string_concat('less ', OutFile, ShellCommand),
                    shell(ShellCommand)
                    ).

print_all_defined_vars:-
                    variable(Name, _, _, _, Value),
                    Value \== undefined,
                    Value \== n,
                    writef('CONFIG_%w=%w\n', [Name, Value]),
                    fail.


print_all_defined_vars.



set_variable_value(Preffix, Suffix, Value):-
            string_concat(Preffix, Suffix, Name),
            set_variable_value( Name, Value).


set_variable_value(Name, Value):-
                    retract(variable(Name, Type, Prompt, Default, _)),
                    assert(variable(Name, Type, Prompt, Default, Value)).


shell_command:-
                    askTermD('Command', 'just kidding', Command),
                    !, Command \==  'just kidding',
                    clear_screen,
                    string_concat(Command, ' 2>&1 |less', Command1),
                    writef('Press "q" to return back\n'), pause,
                    shell(Command1, _).

shell_comamnd.
```

A.2 Config2rules

```
#!/usr/bin/perl
##############################################################################
# NAME
#       $Id: //depot/home/beznosov/Classes/CEN5120/project/linux-kernel/input/config2rules#11$
#
# DESCRIPTION
#       Converts Config.in files in linux kernel source tree
#       (linux/arch/i386/config.in is a root) into a list of rules
#       for futher processing by prolog based configurating
#       expert system.
#
# AUTHOR
#       Konstantin Beznosov (http://www.cs.fiu.edu/~beznosov)
#
# CREATION DATE
#       3/10/97
#
# COPYRIGHT
#       1997 (c) Konstantin Beznosov, All rights Reserved
#
# WARRANTY
#       THIS SOFTWARE IS PROVIDED BY KONSTANTIN BEZNSOOV ‘‘AS IS’’ AND ANY
#       EXPRESSED 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 KONSTANTIN BEZNOSOV OR
#       HIS 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.
#
# LICENSE
#       Can be distributed under GNU General Public License v. 2
#
##############################################################################

##############################################################################
#Options:
#  -b use provided base instead of default one (/usr/src/linux) for
#     recurcive parsing
#  -d produce debug output on stderr
#  -l print redundant dependencies
#  -p produce output suitable for loading directly in Edinberg prolog engine.
#
##############################################################################

require ’getopts.pl’;

Getopts(’db:lp’);

$base = ’/usr/src/linux’ unless $opt_b;

$glob_IF_var_is_on = 0;

$DEBUG = 1 if $opt_d;

if ($opt_p)
{
   &print_header;
   &define_prolog_predicates;
}
```

```
read_config_file(@ARGV[0] ? @ARGV[0] : "/usr/src/linux/arch/i386/config.in", 'fh00');

exit 0;

sub read_config_file {
# parsers config file

   local($filename, $input) = @_;
   $input++;
   print STDERR "Reading \"$filename\"\n" if $DEBUG;

   if ($opt_p)
   {
      print "%-------------------------------------------------------------------\n";
      print "% From file \"$filename\"\n";
      print "%-------------------------------------------------------------------\n";
      print "\n";
   }

   unless(open($input, "<$filename"))
   {
      print STDERR "Can't open file \"@_[0]\": $!\n";
      return;
   }

   while (<$input>) {
      /(hex|int)\s+\'\s*(.*)\'\s+(\w+)\s+(\w+)/
         && print_hex_int_rule($1,$2,$3,$4);

      /bool +\' *(.*)\'\s+CONFIG_(\w+)/
         && print_bool_tristate_rule($1,$2,'b');

      /tristate +\' *(.*)\'\s+CONFIG_(\w+)/
         && print_bool_tristate_rule($1,$2,'t');

      /\s*if\s*\[\s*\"\$CONFIG_(\w+)\"\s*(\!{0,1}=)\s*\"(\w+)\"\s*\]/
         && set_if_var("$1 $2 $3");

      /\s*if\s*\[\s*\!\s+\"\$CONFIG_(\w+)\"\s*(\!{0,1}=)\s*\"(\w+)\"\s*\]/
         && set_if_var("$1 \!$2 $3");

      /\s*if\s*\[\s*\"\$CONFIG_(\w+)\"\s*(\!{0,1}=)\s*\"(\w+)\"\s+\-a\s+\"\$CONFIG_(\w+)\"\s*(\!{0,1}=)\s*\"(\w
         && set_if_var("$1 $2 $3 and \'$4\' $5 $6");

      /\s*if\s*\[\s*\"\$CONFIG_(\w+)\"\s*(\!{0,1}=)\s*\"(\w+)\"\s+\-o\s+\"\$CONFIG_(\w+)\"\s*(\!{0,1}=)\s*\"(\w
         && set_if_var("$1 $2 $3 or \'$4\' $5 $6");

      /^\s*fi/
         && end_if();

      if (/^\s*source\s+([\w\-\/\.]+)/)
      {
         read_config_file("$base/$1", $input);
         if ($opt_p)
         {
            print "%-------------------------------------------------------------------\n";
            print "% Back to file \"$filename\"\n";
            print "%-------------------------------------------------------------------\n";
            print "\n";
         }
      }
   }
   close $input;
```

```
   print STDERR "Back from reading \"$filename\"\n" if $DEBUG;
   return;
}

sub print_bool_tristate_rule {
   if ($opt_p)
   {
      printf "variable(\'@_[1]\', @_[2], \'@_[0]\', n, undefined ).\n";
   }
   else
   {
      printf "%-25.25s|%s| %-50.50s", @_[1], @_[2], @_[0];
   }
   print_dependencies(@_[1]);
}

sub print_hex_int_rule {
   if ($opt_p)
   {
      printf "variable(\'@_[2]\', @_[0], \'@_[1]\', @_[3], undefined ).\n";
   }
   else
   {
      printf "%-25.25s|%1.1s| %-41.41s [%6.6s]", @_[2], @_[0], @_[1], @_[3];
   }
   print_dependencies(@_[2]);
}
sub set_if_var {

   #local @old = grep(/@_[0]/,@glob_IF_vars);
   #print STDERR "@_[0] $#old $#glob_IF_vars @glob_IF_vars\n";
   #return if $#old > 0;


   $glob_IF_var_is_on++;

   if ($opt_p)
   {
      @_[0] =~ s/!=/is_not/g;
      @_[0] =~ s/=/set_to/g;
      @_[0] =~ s/^(\S+)/\'$1\'/; # Sarround variable with quotes
   }

   $glob_IF_vars[$glob_IF_var_is_on] = @_[0];
   #$glob_IF_vars_values[$glob_IF_var_is_on] = "@_[2] @_[1]";

}

sub end_if {
   print STDERR "$glob_IF_var_is_on\n" if $glob_IF_var_is_on < 0;
   $glob_IF_vars[$glob_IF_var_is_on] = "";
   $glob_IF_var_is_on--;
}

sub print_dependencies
{
   local $variable = @_[0];
   local $extra = 0;
   local $i;
   local $j;

   local $no_and = 1;

   if($opt_p)
   {
      print "if \'$variable\' is_wanted";
```

```
   }

   # print dependencies
   for ($i = 1; $i <= $glob_IF_var_is_on ; $i++) {

      # Build a list of all dependencies for this current variable
      $glob_all_vars{$variable} .= " $glob_IF_vars[$i]";

      print STDERR "\nchecking $variable: $glob_all_vars{$variable}\n" if $DEBUG;

      $extra = 0;
      if ( !defined $opt_l ) # produce short dependencies lists
      {
        # Ignore those dependencies that can be infered from any of
        # dependencies before

        # Go through all previous variables and make sure they are
        # not dependent on the current yet
        for ( $j = $glob_IF_var_is_on; $j > $i; $j-- )
        {
            print STDERR $glob_IF_vars[$j],$glob_IF_vars[$i], "\n" if $DEBUG;
            $extra = check4extra($glob_IF_vars[$i],$glob_IF_vars[$j]);
            last if $extra == 1; # do not check futher since there
            # are already some extra dependecies
        }


      }

      next if $extra == 1;
      if ($opt_p)
      {
         printf " and $glob_IF_vars[$i]";
      }
      else
      {
         print " | $glob_IF_vars[$i]";
      }
   }

   if ($opt_p)
   {
      printf " then \'$variable\'.\n";
   }
   else
   {
      print " | $glob_IF_vars[$i]";
   }


   print "\n";
}

sub check4extra
# Checks if new depends on old already.
# check4extra(old,new)
{
   @_[0] =~ /(\w+)/;
   local $old = $1;


   @_[1] =~ /(\w+)/;
   local $new = $1;

   local @list = split(' ',$glob_all_vars{$new});
```

```
   print STDERR "new var $new depends on: ", join(' ',@list),"\n" if $DEBUG;

   local @result = grep(/$old/,@list);
   print STDERR "$new depends on $old:  $glob_all_vars{$new} : $#result\n" if $DEBUG;
   return ($#result >= 0) ? 1 : 0;

}
################################################################################
#
sub define_prolog_predicates
{
   print
"
";


}
################################################################################
#
sub print_header
{
  require 'ctime.pl';
  $today = ctime(time);
print
"% File was generated on $today
";

}
```

B GNU GENERAL PUBLIC LICENSE

GNU GENERAL PUBLIC LICENSE Version 2, June 1991

B.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

B.2 GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to

any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

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

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License. 8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

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

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

END OF TERMS AND CONDITIONS