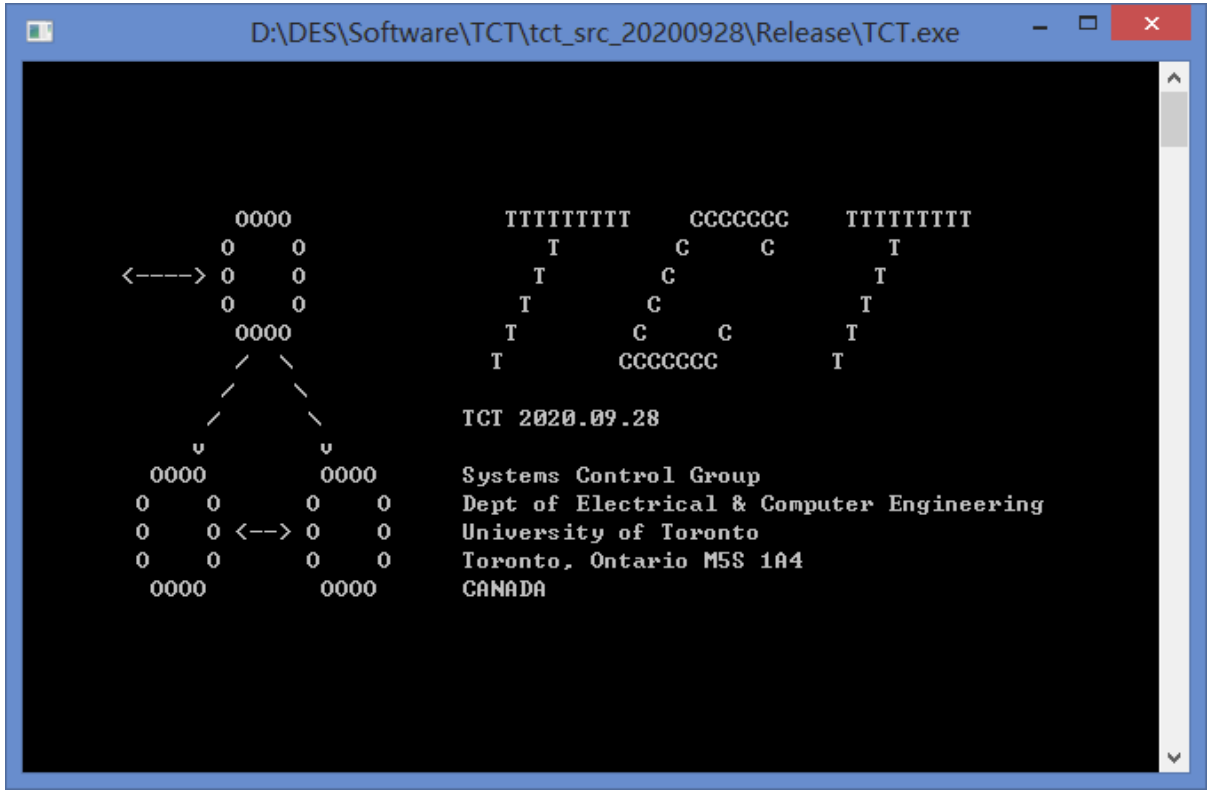
# TCT Development Guide

Renyuan Zhang

2020.10.15

# Main function-Entrance of Procedure




```
int main(int argc, char *argv[])
{
    FILE *in;
    char ch;

    wtct_init(argv[0]);

    if (cmdline) {
        cmdline_tct_run();
        wtct_done();
    } else {
        wtct_run();
        wtct_done();
    }

    return 0;
}
```

Initialize TCT directory

Run TCT with command line; could be called by TCT itself or Matlab

Run TCT

End TCT and return

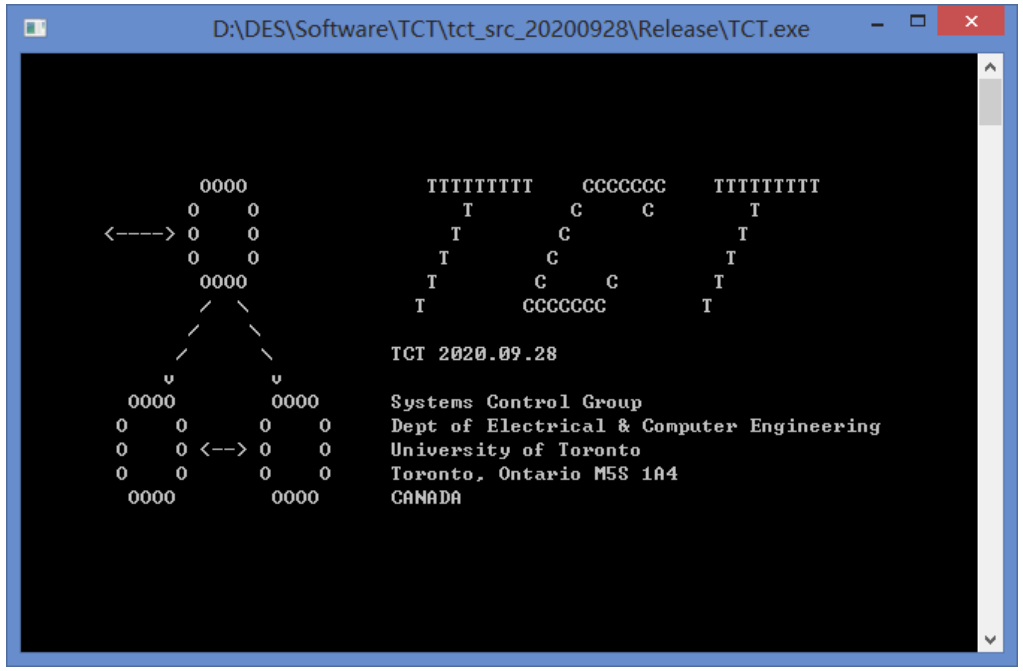
# Section 1:
# Run TCT directly

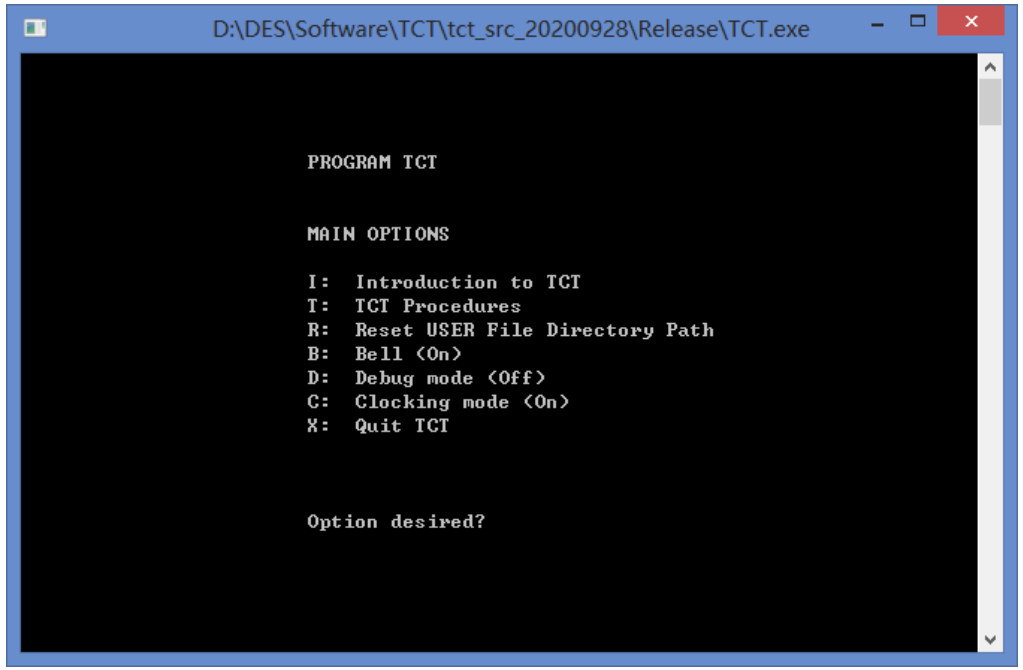
wtct_run()

# Display homepage and main menu

```c
INT_OS wtct_run() {
    char version_str[80];
    char ch;
    INT_OS x,y;

    sprintf(version_str, "%s 2020.09.28", TCTNAME);  /* YY/MM/DD */

    clear();
    tct_logo(version_str);
    move(0,0);
    refresh();
    ch = read_key();

    clear();
    do {
       main_menu();
       refresh();
       x = _wherex();
       y = _wherey();
       do {
           ch = get_command();
           ch = (char) toupper(ch);
           move(y,x);
           refresh();
       } while (strchr(MCommandSet, ch) == NULL);

       process_m_command(ch);
    } while (ch != 'X');

    return 0;
}
```

Process command: I, T, R, B, D, C



# Functions in main menu

```
void process_m_command(char ch)
{
    switch (ch) {
      case 'I': introduction_p();        break;
      case 'R': reset_directory_p();     break;
      case 'B': bell_control_p();        break;
      case 'T': tcommand();              break;
      case 'D': display_control_p();     break;
      case 'O': exit_os_p();             break;
      case 'C': timing_control_p();      break;
   }
}
```

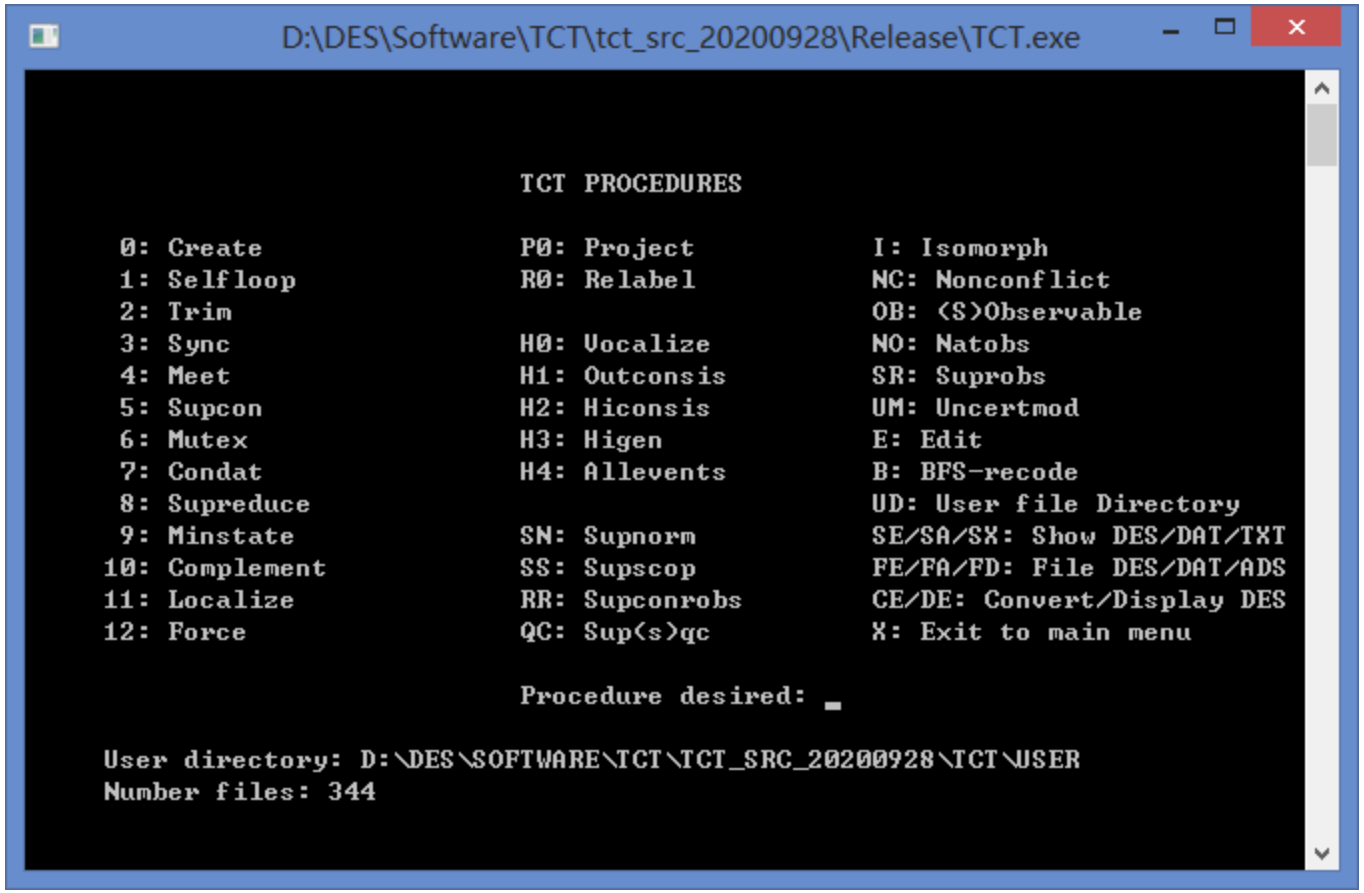
- Display TCT_INFO.PDF (Introduction of TCT procedures)
- Reset (Change) TCT work directory
- Decide whether or not beep when something happened (not used usually)
- Entering into TCT development page
- Decide whether or not print the debug information
- Not used anymore
- Turn on or turn off the time recordings

# TCT Procedures

```
void tcommand()
{
    INT_OS x,y;
    char ch;

    do {
        advance_menu();
        refresh();
        x = _wherex();
        y = _wherey();
        do {
            move(y,x);
            ch = get_upcase_command();
            refresh();
        } while (strchr(CommandSet, ch) == NULL);

        process_command(ch);

    } while (ch != 'X');
}
```

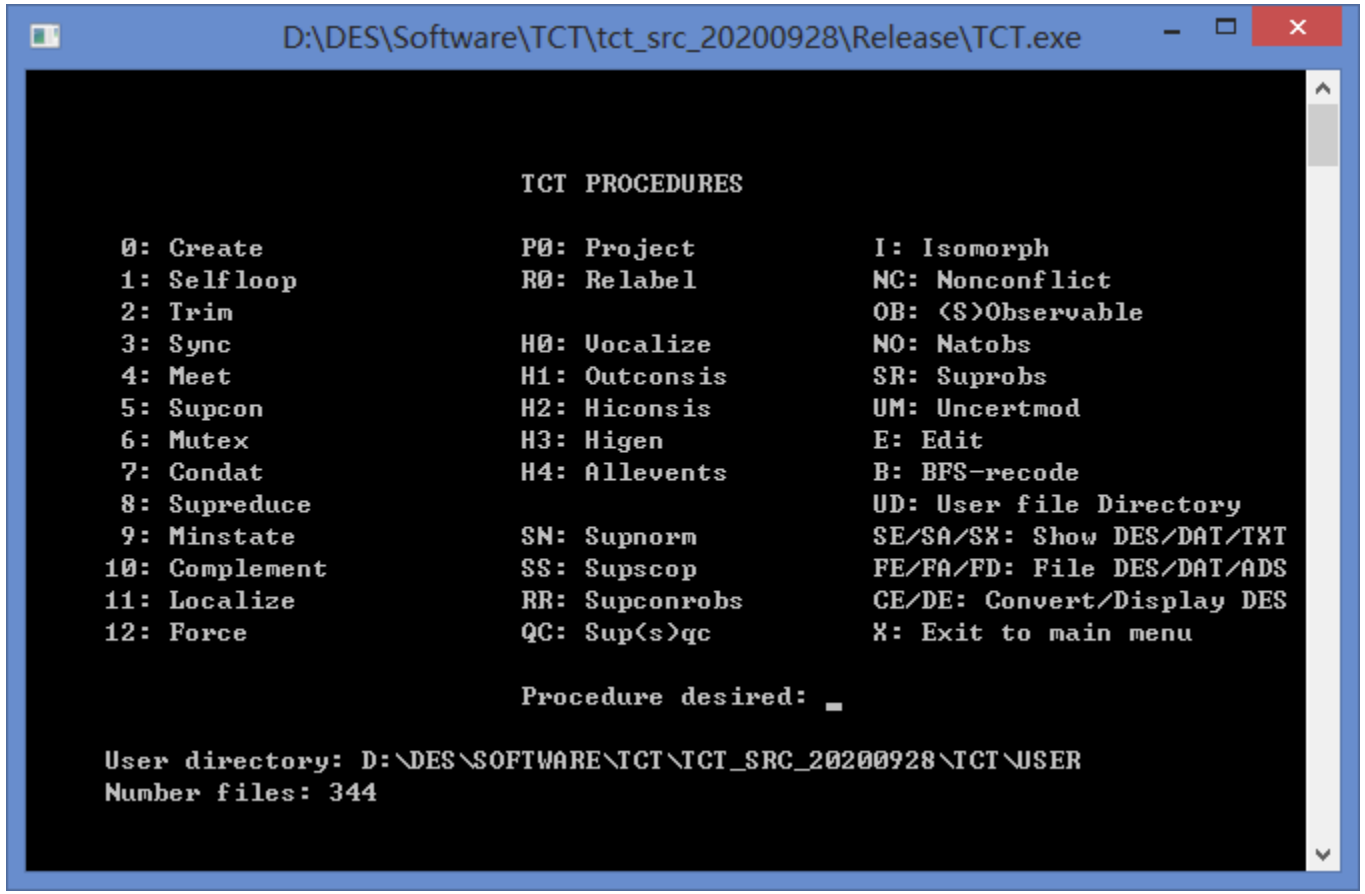
Display the main menu

Process command by 0,1,…



# TCT Procedures

```
void process_command(char ch)
{
  switch (ch) {
     case '0': create_p();        break;
     case '1': _1_proc();         break;
     case '2': trim_p();          break;
     case '3': sync_p();          break;
     case '4': meet_p();          break;
     case '5': supcon_p();        break;
     case '6': mutex_p();         break;
     case '7': condat_p();        break;
     case '8': supreduce_p();     break;
     case '9': minstate_p();      break;
     case 'P': p_proc();          break;
     case 'I': isomorph_p();      break;
     case 'N': n_proc();          break;
     case 'B': bfs_recode_p();    break;
     case 'E': edit_p();          break;
     case 'U': u_proc();          break;
     case 'D': d_proc();          break;
     case 'S': s_proc();          break;
     case 'O': o_proc();          break;
     case 'F': f_proc();          break;
     case 'H': hierchical();      break;
     case 'R': r_proc();          break;
     case 'M': m_proc();          break;
     case 'C': c_proc();          break;
     case 'Q': q_proc();          break;
     case 'Z': z_proc();          break;
   }
}
```



# Example: Supcon (accessed by entering 5)

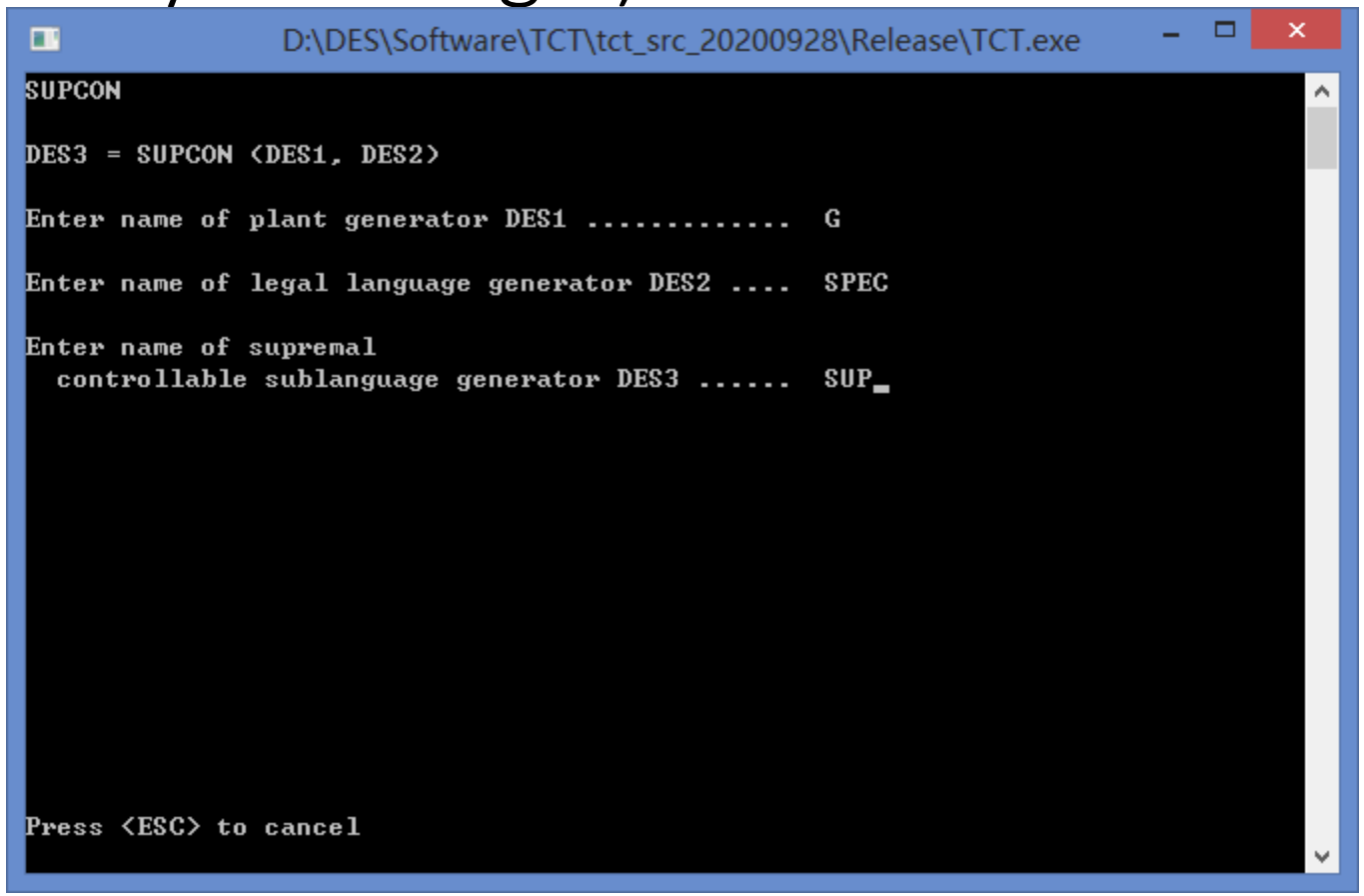
```
void supcon_p()
{
   state_node *t1, *t2, *t3;
   INT_S s1, s2, s3;

   t1 = t2 = t3 = NULL;
   s1 = s2 = s3 = 0;

   supcon_r(&t1, &s1, &t2, &s2, &t3, &s3);

   if (mem_result == 1) {
      mem_result = 0;
      OutOfMemoryMsg();
      user_pause();
   } else {
      if (!quit) {
        supcon_makeit(t3, s3);
        user_pause();
      }
   }

   echo_free();
   freedes(s1, &t1);
   freedes(s2, &t2);
   freedes(s3, &t3);
}
```

Main procedure of supcon

Write the computing result into the makeit.txt file

# Example: Supcon (accessed by entering 5)

```
void supcon_r(state_node **t1,
              INT_S *s1,
              state_node **t2,
              INT_S *s2,
              state_node **t3,
              INT_S *s3)
{
   INT_S  init;
   INT_S  *macro_ab, *macro_c;
   INT_OS    result;

   macro_ab = NULL;  macro_c  = NULL;

   clear();
   supcon_header();
   quit = getname("Enter name of plant generator DES1 .............  ",
                  EXT_DES, name1, false);
   if (quit) return;

   quit = getname("Enter name of legal language generator DES2 ....  ",
                  EXT_DES, name2, false);
   if (quit) return;

   printw("Enter name of supremal"); println();
   quit = getname("  controllable sublanguage generator DES3 ......  ",
                   EXT_DES, name3, true);
   if (quit) return;

   move(22,0); clrtoeol();
   move(23,0); clrtoeol();
   printw("Processing:  Please wait...");
   refresh();

   /* Pass to command line version of this program */
   mark_start_time();
   result = supcon_runProgram(name3, name1, name2);
   mark_stop_time();
```




Record the starting time

Call TCT with command line to get the result of SUPCON

Recording the stop time

# Example: Supcon (accessed by entering 5)

```
INT_OS supcon_runProgram(char *name3, char *name1, char *name2)
{
    FILE *f1;

    f1 = fopen(get_prm_file(), "w");
    if (f1 == NULL)
       return -1;  /* Some type of system error */

    fprintf(f1, "%d\n", debug_mode); /* Debug mode */
    fprintf(f1, "%d\n", minflag);
    fprintf(f1, "5\n");  /* SUPCON */
    fprintf(f1, "%s\n", name1);
    fprintf(f1, "%s\n", name2);
    fprintf(f1, "%s\n", name3);
    fclose(f1);

    //cmdline_tct_run();
    return mySystem();
}
```

```
INT_OS mySystem()
{
    char runParm[256];
    INT_OS result;

    sprintf(runParm, "\"%s\" -cmdlin

    if (debug_mode == 0)
    {
       open_stdout();
       open_stderr();
    }

    result = system(runParm);

    if (debug_mode == 0)
    {
       close_stdout();
       close_stderr();
    }

    if (result != 0)
       return -2;  /* Some type of system error */

    return get_ctct_result();
}
```

system(runParam) is a system function. It implements the operations "runParam" in cmd.exe. Here, runParam = "TCT.exe -cmdline".
By this approach, we can call TCT in other procedures (e.g. MATLAB).

To call TCT with command line, the parameters are passed by a file called ctct.prm (containing the necessary information). For SUPCON, the parameters are name1 (plant), name2 (spec), and name3 (sup).

Get TCT result of executing the above command.

# Section 2:
# Run TCT with command line

cmdline_tct_run()

# Functions called by TCT command line

```
INT_OS cmdline_tct_run(void) {
    FILE *f1;
    INT_OS oper;

    f1 = fopen(get_prm_file(), "r");
    if (f1 == NULL)
    {
       ctct_result(CR_NO_PRM_FILE);
       return -1;
    }

    fscanf(f1, "%d\n", &debug_mode);
    fscanf(f1, "%d\n", &minflag);
    fscanf(f1, "%d\n", &oper);
    switch (oper) {
    case 0 :  create_program(f1);          break;
    case 1 :  selfloop_program(f1);        break;
    case 2 :  trim_program(f1);            break;
    case 3 :  synck_program(f1);           break;
    case 4:   meetk_program(f1);           break;
    case 5 :  supcon_program(f1);
    case 6 :  mutex_program(f1);           break;
    case 7 :  condat_program(f1);          break;
    case 8 :  supreduce_program(f1);       break;
    case 9 :  minstate_program(f1);        break;
    case 10:  complement_program(f1);      break;
    case 11:  localize_program(f1);        break;
    case 12:  force_program(f1);           break;
    case 13:  project_program(f1);         break;
    case 14:  convert_program(f1);         break;
    case 15:  vocalize_program(f1);        break;
    case 16:  outconsis_program(f1);       break;
    case 17:  hiconsis_program(f1);        break;
    case 18:  higen_program(f1);           break;
    case 19:  allevents_program(f1);       break;
    case 20:  supnorm_program(f1);         break;
    case 21:  supscop_program(f1);         break;
```

TCT functions will be executed according to the code “oper” in the ctct.prm file.

The code “oper” for supcon is 5, and thus the function supcon_program(f1) will be executed.

# Functions called by TCT command line

```c
void supcon_program(FILE *f1)
{
    state_node *t1, *t2, *t3;
    INT_S s1, s2, s3, init;
    INT_S *macro_ab, *macro_c;

    macro_ab = NULL;  macro_c  = NULL;
    t1 = t2 = t3= NULL;
    s1 = s2 = s3 = 0;

    /* Use "fgets" as names could have spaces in it */
    if (fgets(name1, MAX_FILENAME, f1) == NULL)
    {
        fclose(f1);
        remove(prm_file);
        ctct_result(CR_PRM_ERR);
        exit(0);
    }
    name1[strlen(name1)-1] = '\0';

    if (fgets(name2, MAX_FILENAME, f1) == NULL)
    {
        fclose(f1);
        remove(prm_file);
        ctct_result(CR_PRM_ERR);
        exit(0);
    }
    name2[strlen(name2)-1] = '\0';

    if (fgets(name3, MAX_FILENAME, f1) == NULL)
    {
        fclose(f1);
        remove(prm_file);
        ctct_result(CR_PRM_ERR);
        exit(0);
    }
    name3[strlen(name3)-1] = '\0';
```

Get name1, name2, and name3 from the ctct.prm file

```c
    init = 0L;
    getdes(name1, &s1, &init, &t1);
    getdes(name2, &s2, &init, &t2);

    meet2(s1,t1,s2,t2,&s3,&t3,&macro_ab,&macro_c);
    freedes(s2,&t2); t2 = NULL;
    trim2(&s3,&t3,macro_c);
    shave1(s1,t1,&s3,&t3,macro_c);

    if (mem_result != 1)
    {
       filedes(name3, s3, init, t3);
    }
    else
    {
        ctct_result(CR_OUT_OF_MEMOR
        exit(0);
    }
```

Core implementation of Supcon procedure

Store the result/ or output an error.

# DES data structure in TCT

```
typedef struct state_node {
    INT_B  marked  : 1;
    INT_B  reached : 1;
    INT_B  coreach : 1;
    INT_V vocal : 13;
    INT_T numelts;
    tran_node *next;
} state_node;

typedef struct tran_node {
     INT_S data2:22;
     INT_T data1:10;
} tran_node;
```

a state is marked or not

a state is reached or not

a state is coreachable or not

a state is vocal or not

number of transitions

pointer to transition structure

target state of a transition

event label of a transition

```
INT_OS filedes(char*        name,
               INT_S        size,
               INT_S        init,
               state_node *data)
```

Store a DES into a file

```
INT_B  getdes(char*         name,
              INT_S         *size,
              INT_S         *init,
              state_node **data)
```

Get a DES from a file

Definition of DES

```
state_node *t1, *t2, *t3;
INT_S s1, s2, s3, init;


init = 0L;
getdes(name1, &s1, &init, &t1);
getdes(name2, &s2, &init, &t2);

meet2(s1,t1,s2,t2,&s3,&t3,&macro_ab,&macro_c);
freedes(s2,&t2); t2 = NULL;
trim2(&s3,&t3,macro_c);
shave1(s1,t1,&s3,&t3,macro_c);

if (mem_result != 1)
{
   filedes(name3, s3, init, t3);
}
else
{
    ctct_result(CR_OUT_OF_MEMORY);
    exit(0);
}
```

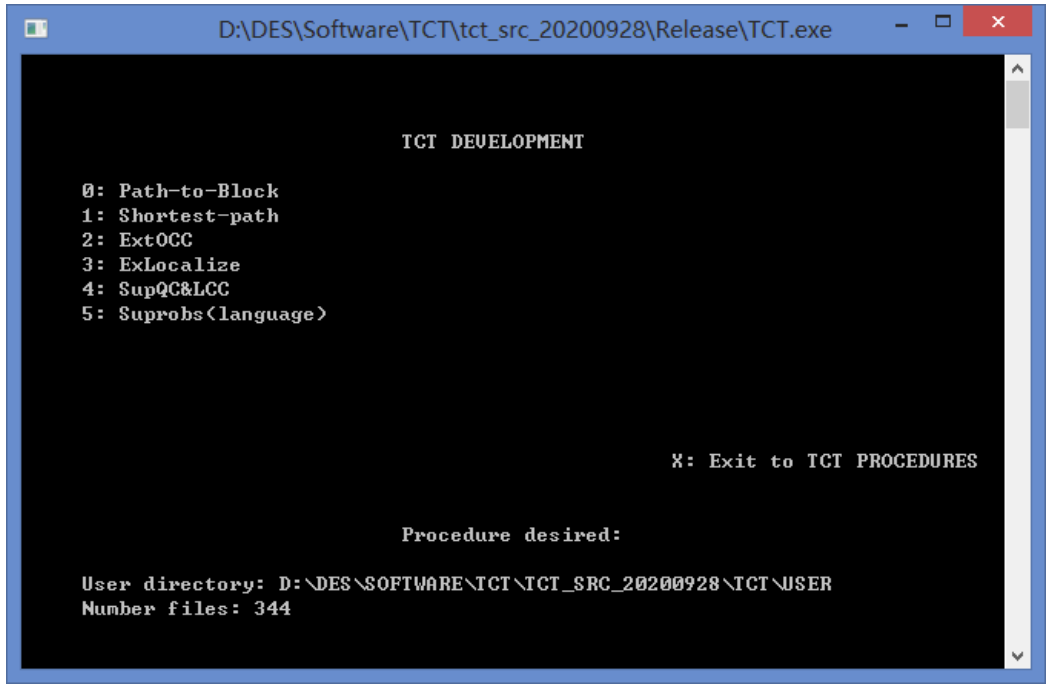
# Section 3:
# Add new procedure to TCT

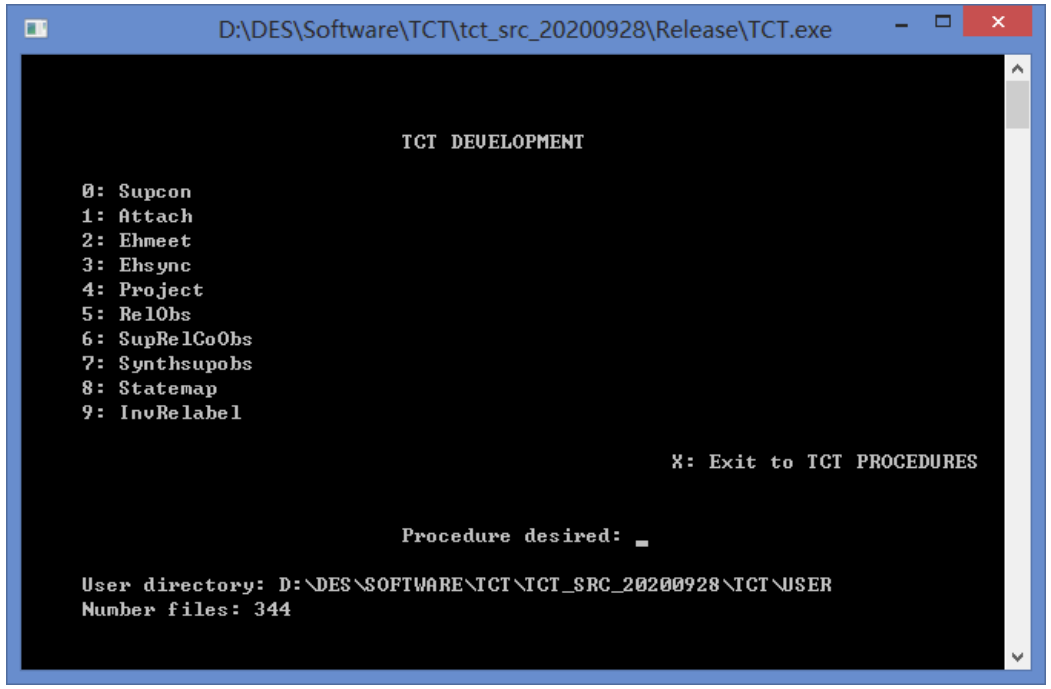
# Development pages

```
void process_command(char ch)
{
  switch (ch) {
     case '0': create_p();        break;
     case '1': _1_proc();         break;
     case '2': trim_p();          break;
     case '3': sync_p();          break;
     case '4': meet_p();          break;
     case '5': supcon_p();        break;
     case '6': mutex_p();         break;
     case '7': condat_p();        break;
     case '8': supreduce_p();     break;
     case '9': minstate_p();      break;
     case 'P': p_proc();          break;
     case 'I': isomorph_p();      break;
     case 'N': n_proc();          break;
     case 'B': bfs_recode_p();    break;
     case 'E': edit_p();          break;
     case 'U': u_proc();          break;
     case 'D': d_proc();          break;
     case 'S': s_proc();          break;
     case 'O': o_proc();          break;
     case 'F': f_proc();          break;
     case 'H': hierchical();      break;
     case 'R': r_proc();          break;
     case 'M': m_proc();          break;
     case 'C': c_proc();          break;
     case 'Q': q_proc();          break;
     case 'Z': z_proc();          break;
   }
}
```

Z1 (Useful and tested procedures)

Z2 (open for users)



# Development pages

```c
void ctct_dev_ext2()
{
    INT_OS x,y;
    char ch;

    do {
        ctct_menu_page2();
        refresh();
        x = _wherex();
        y = _wherey();
        do {
            move(y,x);
            ch = get_upcase_command();
            refresh();
        } while (strchr(DCommandSet, ch) == NULL);

        dcommand_ext2(ch);

    } while (ch != 'X');
}
```

```c
void dcommand_ext2(char ch)
{
    switch (ch) {
        // Left panel
    case '0': supcon_comb_p();          break;
    case '1': attach_p();               break;
    case '2': ehmeet_p();               break;
    case '3': ehsync_p();               break;
    case '4': plain_project_p();        break;
    case '5': rel_observ_p();           break;
    case '6': sup_rel_coobs_p();        break;
    case '7': synsupobs_p();            break;
    case '8': statemap_p();             break;
    case '9': inv_relabel_p();          break;

        // Right pannel
```

```c
void ctct_menu_page2() {
    INT_OS xLeft;    /* left column offset   */
    INT_OS xMiddle;  /* middle column offset */
    INT_OS xRight;   /* right column offset  */
    INT_OS y;        /* row offset           */
    char dir[_MAX_PATH];
    INT_OS num_entries;
    struct _finddata_t *namelist;

    xLeft   = 5;
    xMiddle = 31;
    xRight  = 53;
    y       = 4;

    clear();
    move(3   ,xMiddle);  printw("%s DEVELOPMENT", TCTNAME);

    /* Left command list panel */
    move(y+1, xLeft);   addstr("0: Supcon");
    move(y+2 ,xLeft);   addstr("1: Attach");
    move(y+3 ,xLeft);   addstr("2: Ehmeet");
    move(y+4 ,xLeft);   addstr("3: Ehsync");
    move(y+5, xLeft);   addstr("4: Project");
    move(y+6, xLeft);   addstr("5: RelObs");
    move(y+7, xLeft);   addstr("6: SupRelCoObs");
    move(y+8, xLeft);   addstr("7: Synthsupobs");
    move(y+9, xLeft);   addstr("8: Statemap");
    move(y+10,xLeft);   addstr("9: InvRelabel");

    move(y+12,xRight);  addstr("X: Exit to TCT PROCEDURES");
```

Example: Add a new function named “TEST” accessed by “T”.

(1) Add one line “move(y+1, xMiddle); addstr(“T: TEST”)” in ctct_menu_page2();

(2) Add one line “case ‘T’: test_p()” in dcommand_ext2();

(3) Write your procedure test_p().

# Section 4:
# Add new TCT calling interface to Matlab

# Call TCT by matlab

```c
int main(int argc, char *argv[])
{
    FILE *in;
    char ch;


    wtct_init(argv[0]);

    if (cmdline) {
        cmdline_tct_run();
        wtct_done();
    } else {
        wtct_run();
        wtct_done();
    }

    return 0;
}
```

Run TCT with command line. Thus, TCT could be called by TCT itself or Matlab

```c
INT_OS cmdline_tct_run(void) {
    FILE *f1;
    INT_OS oper;

    f1 = fopen(get_prm_file(), "r");
    if (f1 == NULL)
    {
        ctct_result(CR_NO_PRM_FILE);
        return -1;
    }

    fscanf(f1, "%d\n", &debug_mode);
    fscanf(f1, "%d\n", &minflag);
    fscanf(f1, "%d\n", &oper);
    switch (oper) {
    case 0 :  create_program(f1);          break;
    case 1 :  selfloop_program(f1);        break;
    case 2 :  trim_program(f1);            break;
    case 3 :  synck_program(f1);           break;
    case 4:   meetk_program(f1);           break;
    case 5 :  supcon_program(f1);          break;
    case 6 :  mutex_program(f1);           break;
    case 7 :  condat_program(f1);          break;
    case 8 :  supreduce_program(f1);       break;
    case 9 :  minstate_program(f1);        break;
    case 10:  complement_program(f1);      break;
    case 11:  localize_program(f1);        break;
    case 12:  force_program(f1);           break;
    case 13:  project_program(f1);         break;
    case 14:  convert_program(f1);         break;
    case 15:  vocalize_program(f1);        break;
    case 16:  outconsis_program(f1);       break;
    case 17:  hiconsis_program(f1);        break;
    case 18:  higen_program(f1);           break;
    case 19:  allevents_program(f1);       break;
    case 20:  supnorm_program(f1);         break;
    case 21:  supscop_program(f1);         break;
```

# Call TCT by matlab

```
Q1 = 0;
Qm1 = [];
Tran1 = [];
create('IESI_PLANT1', Q1, Tran1, Qm1);

Q2 = 0;
Qm2 = [];
Tran2 = [];
create('IESI_SPEC1', Q2, Tran2, Qm2);

supcon('IESI_SUP1', 'IESI_PLANT1', 'IESI_SPEC1');
```

Call TCT procedures create_program to create a plant DES and a spec DES

Call TCT procedures supcon_program to compute the supervisor for given plant and spec DES

# supcon in Matlab

```
function supcon(sup, plant, spec)
global path;
global tct_name;
global prm_file;
global rst_file;
global err_info;

if err_info(2) ~= 0
    return;
end

code = 5;
err_info(1) = code;

% check if the input DES plant exists
full_name = strcat(path, '\');
full_name = strcat(full_name, plant);
full_name = strcat(full_name, '.DES');

if ~exist(full_name, 'file')
    err_info(2) = 7;
    printerror(plant);
    return;
end
```

Declare function supcon with parameters sup, plant and spec

Generate full path of plant DES, and check if the file exists.

# supcon in Matlab

```
% check if the input DES spec exists
full_name = strcat(path, '\');
full_name = strcat(full_name, spec);
full_name = strcat(full_name, '.DES');

if ~exist(full_name, 'file')
    err_info(2) = 7;
    printerror(spec);
    return;
end
```

Generate full path of spec DES, and check if the file exists.

```
fid = fopen(prm_file, 'w');
if fid == -1 % cannot open the specified file
    err_info(2) = 1;
    printerror(prm_file);
    return;
end

fprintf(fid, '0\n'); % debug_mode
fprintf(fid, '1\n'); % min_flag
fprintf(fid, '%d\n', code); % code for supcon
fprintf(fid, '%s\n', plant); % name of plant
fprintf(fid, '%s\n', spec); % name of specification
fprintf(fid, '%s\n', sup); % name of supervisor

fclose(fid);
```

Generate ctct.prm file for calling TCT procedures. Note here that the format of ctct.prm must be consistent with the one (supcon_runProgram) defined in TCT

# supcon in Matlab

```
cmdline = strcat(tct_name, ' -cmdline');
if system(cmdline) ~= 0
    err_info(2) = 2;
    printerror;
    return;
end
```

Call TCT with operation “-cmdline”, as done by function mySystem() in TCT.

```
full_name = strcat(path, '\');
full_name = strcat(full_name, sup);
full_name = strcat(full_name, '.DES');

if ~exist(full_name, 'file')
    err_info(2) = 4;
    printerror(sup);
    return;
end

printerror;

[state_size, tran_size] = getdes_parameter(sup);
```

Check if the result (sup) has been produced, and get the state and transition numbers of the result DES.

# supcon in Matlab

```
fid = fopen('tmp.$$$', 'w');
if fid == -1 % cannot open the specified file
    err_info(2) = 1;
    printerror('tmp.$$$');
    return;
end
fprintf(fid, '%s = Supcon(%s,%s)', sup, plant, spec);
fprintf(fid, '   (%d,%d)\n\n', state_size, tran_size);
fclose(fid);

mergechop(length(sup) + 3);



fclose('all'); % close all open files
if exist(prm_file, 'file')
    delete(prm_file);
end
if exist(rst_file, 'file')
    delete(rst_file);
end

end %function
```

Update the makeit.txt file

Finally remove the temporary files

# Section 5:
# Matlab functions compatible with DES in TCT

# Matlab functions compatible with TCT

To operate the DES files in TCT, we need to know how to write a DES structure (in matlab) into a TCT DES file, and read a DES structure from a TCT DES file.

- Currently, we only have create function in matlab to write a DES with the same format in TCT.
- Thus, we need a matlab function to get the DES information from a DES file in TCT (like getdes() in TCT).

# create (calling create_program in TCT)

```c
void create_program(FILE *f1)
{
    //INT_T slist, *list;
    state_node *t; INT_S s, init;
    INT_OS ee, flag;
    INT_S i,j,k;
    INT_B ok;


    t = NULL; s = 0;

    /* Use "fgets" as names could have spaces in it */
    if (fgets(name1, MAX_FILENAME, f1) == NULL)
    {
        fclose(f1);
        remove(prm_file);
        ctct_result(CR_PRM_ERR);
        exit(0);
    }
    name1[strlen(name1)-1] = '\0';

    fscanf(f1, "%d\n", &s);

    t = newdes(s);

    while( fscanf(f1, "%d" , &ee) != EOF)
    {
        if(ee == -1){
            break;
        }
        t[ee].marked = true;
    }
```

Read states number

Read marker states

```c
    flag = 0;
    while( fscanf(f1, "%d" , &ee) != EOF)
    {
        if(flag == 0){
            flag = 1;
            i = ee;
            continue;
        }else if(flag == 1){
            flag = 2;
            j = ee;
            continue;
        }else if(flag == 2){
            flag = 0;
            k = ee;
            addordlist1((INT_T)j,k,&t[i].next, t[i].numelts, &ok);
            if(ok) t[i].numelts ++;
            continue;
        }
    }

    fclose(f1);
    remove(prm_file);

    init = 0L;
    filedes(name1, s, init, t);

    if(mem_result == 1) {
        ctct_result(CR_OUT_OF_MEMORY);
        exit(0);
    }
    freedes(s, &t);
}
```

Read transitions

Store obtained DES

A matlab function getdes can be implemented similarly.

# The End!