CSCI 4010.002

DESIGN DOCUMENT

Monitor Program for Computer Labs

Developed by:
Carol C. Stimson
October 15, 1990

[Handwritten note: nice level of detail for implementation]
Please note that this is only right right-side page of the Hierarchy Chart. The student's document included additional material, but you can get the idea based on what is shown here.
Constants:

maxsystems = 10;
maxstations = 60;

Enumerated Types:

termtype = (karrowup, karrowdown, karrowleft, karrowright,
            kselect, kescape, kbackspace, kasciichars, kf1,
            kf2, kf3, kf4, kf5, kf6, kf7, kf8, kf9, kf10,
            kmainmenu, kuser, korderby, kwatinglist, kup,
            kdown, kreserve, kadduser, kcanceluser, klistuser,
            kswitchstations, khelp, kexit);

statustype = (up, down, reserve);

orderbytype = (name, number, time);

actiontype = (add, delete, change, switch, display);

Miscellaneous Types:

string11 = string[11];
string20 = string[20];
string2 = string[2];
string40 = string[40];
charset = set of char;
termset = set of termtype;
Systemptr = Pointer to SystemRec;
SystemConfigptr = Pointer to SystemConfigRec;

Record Types:

WaitingListRec = record
    WL_name : string20;
    WL_SSN : string11;
end;

StationRec = record
    stationstatus : statustype;
    student : string20;
    SSN : string11;
    station_number : string2;
    items_checked_out : string40;
    time_logged_in : integer;
end;
SystemRec = record
  station : array[1..99] of StationRec;
  WaitingList : array[1..60] of WaitingListRec;
  numinuse : integer;
  numavailable : integer;
  numwaiting : integer;
  stationorder : array[1..60] of integer;
  nameorder : array[1..60] of integer;
  timeorder : array[1..60] of integer;
end;

SystemConfigRec = record
  systemname : string[20];
  startingstation : integer;
  endingstation : integer;
  numdown : integer;
  nuareserved : integer;
end;

ConfigRec = record
  numsystems : integer; (0-10)
  pword : string[5]; (manager's password)
  backuptime : integer; (minutes between saves)
  stattime : integer; (minutes between saves)
  backuppath : string; (pathname of backup file)
  statpath : string; (pathname of statistics file)
  MODpath : string; (pathname of message of day file)
  colorconfig : array[1..30] of integer;
  (items assigned to colors)
end;

Variant1Rec = record
  case integer of
  1 : (rec1 : SystemRec);
  2 : (EDFmsg : string);
end;

Variant2Rec = record
  case integer of
  1 : (firstrec : ConfigRec);
  2 : (otherrec : SystemConfigRec);
end;
### System Recs

```
<table>
<thead>
<tr>
<th>Station</th>
<th>Wait List</th>
<th>Num In Use</th>
<th>Num Avail</th>
<th>Num Waiting</th>
<th>Station Ord</th>
<th>Name Ord</th>
<th>Time Ord</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
Dynamically Allocated as Needed
```

An array of pointers to System Rec

### Each System Rec:

```
<table>
<thead>
<tr>
<th>Station</th>
<th>Wait List</th>
<th>Num In Use</th>
<th>Num Avail</th>
<th>Num Waiting</th>
<th>Station Ord</th>
<th>Name Ord</th>
<th>Time Ord</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Array of Station Rec, Array of Integer, Array of Integer, Array of Integer

### Station Rec

<table>
<thead>
<tr>
<th>station status</th>
<th>student</th>
<th>SSN</th>
<th>station number</th>
<th>items checked out</th>
<th>time logged in</th>
</tr>
</thead>
</table>

### Waiting List Rec

<table>
<thead>
<tr>
<th>WL_name</th>
<th>WL_SSN</th>
</tr>
</thead>
</table>
SystemConfig

SystemConfigRecs

Dynamically Allocated as Needed

Array of Pointers to SystemConfigRec

One SystemConfigRec exists for each system defined.

ConfigRec

One ConfigRec exists for the entire Monitor Program.

Array of Integer
Global Variables:

currenttime : integer;
acceptablechars : charset;
acceptableterm : termset;
backup : array[1..2] of string;
system : array[1..10] of Systemptr;
config : array[1..10] of SystemConfigptr;
newsystem : Systemptr;
newconfig : SystemConfigptr;
term : teratype;
order : orderbytype;
status : statustype;
action : actiontype;
backup1, backup2 : file of Variant1Rec;
configurationfile : file of Variant2Rec;
variant1 : Variant1Rec;
variant2 : Variant2Rec;
Statistics File:

The Statistics File is a standard ASCII text file. The following fields define one line of the Statistics File. The fields are separated by commas. One line containing this data is output to the Statistics file for each existing system. This is done each time the statistics are saved. One statistics file is generated each month and contains all the statistics for all of the systems.

Day : integer
Month : integer
Year : integer
Hour : integer
Minute : integer
SystemName : string20
NumInuse : integer
NumWaiting : integer
loggedIn : integer (since last Statistics were saved)
loggedout : integer (since last Statistics were saved)

Configuration File:

The Configuration file consists of one record containing the basic configuration followed by one record for each existing system containing that system's configuration. This is handled through the use of Variant2Rec. This file is re-written each time the manager menu is exited. This file is also re-written if the number of stations up, down or reserved changes. An array of pointers to the SystemConfigRec is used.

firstRec : ConfigRec (basic configuration info)
from 1 to 10 of this type record:
otherRec : SystemConfigRec (contains each system's configuration information)

See Data Structures section for a complete description of ConfigRec, SystemConfigRec, Variant2Rec & SystemConfig.
Backup Files:

An array of 2 backup filenames exists: 'backup1' and 'backup2'. The file to be written alternates between them. This is done in case a system crash occurs during an actual 'write' and corrupts the file. This way there is a guarantee that at least one of the backup files exists.

If the operator 'quits' the system, these backup files are deleted. In this manner, the system can tell at the time it is brought up if it is recovering from a crash (the backup file exists) or if it is a normal startup (no backup file exists).

The backup file consists of a record for each existing system followed by an end of file record to check the integrity of the backup file. This is handled through the use of Variant1Rec. An array of pointers to the SystemRec is used.

From 1 to 10 of this type records:

recl : variant record of type SystemRec; (contains the station, waiting list, & other system information for each system)

Followed by:

EOFmsg : variant record of type string

See Data Structures section for a complete description of SystemRec, Variant1Rec, and system.

Message of the Day File:

This file is defined elsewhere by the manager. The manager gives the program the pathname of where this file is located. This file contains up to 255 characters in standard ASCII format. The manager may type multiple lines in the file. These lines are automatically concatenated together. If more than 255 characters exist in the file, any over 255 are discarded.
Main Program
Init;
Process_keys;

Procedure Init
initialize variables
Order := number;
Read_Files;
Draw_Opening_Screen;
Get_Key; (Wait for any key to be pressed)
close window
Draw_Main_Screen; (Draws the Main Monitor Screen)

Procedure Process_keys
Get_Key;
Case Key of
  kfl..kfl0: Function keys;
  kmainmenu..kexit: Hotkeys;
end;

Procedure Read_Files
turn off I/O checking and attempt to open configuration file
if configuration file exists
  turn on I/O checking
  read configuration file
turn off I/O checking and attempt to open backup files
if backup files exists
  turn on I/O checking
  if time(backup1) < time(backup2) AND
    filesize(backup1) < filesize(backup2) then
    filenum := 1
  else
    if time(backup1) > time(backup2) AND
      filesize(backup1) > filesize(backup2) then
      filenum := 2
      give warning( 'Both backup files are Corrupt' ) & get out
    open & read last record of backup[filename]
    if corrupt then
      filenum := 3 - filenum
      open and read last record of backup[filename]
    if not corrupt then read the other records
    otherwise
      read the other records
    open & read message of day file
display message of day
close backup file, configuration file, & messageofday file
else
  manager
Procedure Draw_Opening_Screen
  clear the screen
  open window with endtitle 'Press any Key to Continue'
  write 'Welcome to the Computer Laboratory Monitoring System'
  write version number of the program

Procedure Draw_Main_Screen (receives system number, order type)
  clear the screen
  write headings (system name, number available, number in use,
               number waiting, number down, number reserved,
               St, Tm, Name)
  case order of
    name : begin
      for i := 1 to numinuse do
        begin
          Calculate row and column
          sprint(NameOrder[i]);
        end
    end
  number : begin
    for i := 1 to numinuse do
      begin
        Calculate row and column
        sprint(StationOrder[i]);
      end
  end
  time : begin
    for i := 1 to numinuse do
      begin
        Calculate row and column
        sprint(TimeOrder[i]);
      end
  end
end case
procedure Get_Key (return ch and term)
    while a key is not pressed do
        get date & time & display them
        if ElapsedTimeSince the message file was read > msgtime then
            read Message of Day File
        display Message of Day
        convert current time into minutes
        if ElapsedTimeSince the crash file was written > crashtime
            then WriteFile (backup)
        if ElapsedTimeSince the statistics file was written >
            stattime then WriteFile (stat)
        read the key and store in ch
        if it is an extended key, (ord(ch) = 0) then
            read key again and store in ch
            case (ord(ch)) of
            16 : term := kexit;
            17 : term := kwatinglist;
            19 : term := kreserve;
            23 : term := kuser;
            24 : term := korderby;
            30 : term := kadduser;
            31 : term := kswitchstations;
            32 : term := kdown;
            35 : term := khelp;
            38 : term := klistuser;
            45 : term := kexit;
            46 : term := kcaneluser;
            50 : term := kmainmenu;
            59 : term := kf1;
            60 : term := kf2;
            61 : term := kf3;
            62 : term := kf4;
            63 : term := kf5;
            64 : term := kf6;
            65 : term := kf7;
            66 : term := kf8;
            67 : term := kf9;
            68 : term := kf10;
            72 : term := kuparrow;
            75 : term := kleftarrow;
            77 : term := krightarrow;
            80 : term := kdownarrow;
        end
    else if it's not an extended key
        case (ord(ch)) of
            8 : term := kbackspace
            32..126 : term := kasciichr
        end
    end
Procedure FunctionKeys
  case term of
    kf1 : systemnum := 1;
    kf2 : systemnum := 2;
    kf3 : systemnum := 3;
    kf4 : systemnum := 4;
    kf5 : systemnum := 5;
    kf6 : systemnum := 6;
    kf7 : systemnum := 7;
    kf8 : systemnum := 8;
    kf9 : systemnum := 9;
    kf10 : system_num := 10;
    if systemnum > numsystems then
      warning('There is no system defined for this Key')
      systemnum := 1;
      Draw_Main_Screen; (pass order type & system number)
  end

Procedure HotKeys
  case term of
    kmainmenu : mainmenu;
    kuser : user;
    korderby : orderby;
    kwaitinglist : waitinglist;
    kdown : down;
    kreserve : reserve;
    kadduser : adduser;
    kcanceluser : canceluser;
    klistuser : listuser;
    kswitchstations : switchstations;
    khelp : help;
    kexit : exit;
  end

Function ElapsedTimeSince (receives lasttime : integer) : integer
  ElapsedTimeSince := Currenttime - lasttime;
Procedure Write_file (receives filename)
    if filename = 'backup' then
        begin
            filename := 3 - filename; (toggle to other backup file)
            open backup[filename]
            for i := 1 to numsystems do
                variant1.rec1 := system pointed to by system[i]
                write variant1 to backup[filename]
                variant.EOFmsg := 'This is the end of the backup file!!!'
                write variant1 to backup[filename]
            close backup[filename]
        end
    if filename = 'stat' then
        begin
            filename := 'stat' + month + year
            open filename
            for i := 1 to numsystems do
                Build_Statline(pass i)
                write the new statline
                loggedin := 0;
                loggedout := 0;
            close stat
        end
    if filename = 'config' then
        begin
            open configuration file
            variant2.firstrec := systemconfig
            write variant2 to config file
            for i := 1 to numsystems do
                variant2.otherrec := what config[i] is pointing to
                write variant2 to config file
            close configuration file
        end

Procedure mainmenu
    Make the horizontal Menu
    case currentpos of
        1 : user;
        2 : orderby;
        3 : waitinglist;
        4 : system;
        5 : down;
        6 : reserve;
        7 : manager;
        8 : exit;
    else
        close window
    end
Procedure user
   Make a vertical Menu
   case currentpos of
      1 : adduser;
      2 : canceluser;
      3 : listuser;
      4 : switchstations;
   else
      close window;

Procedure orderby
   Make a vertical Menu
   case currentpos of
      1 : Draw_Main_Screen(name);
      2 : Draw_Main_Screen(number);
      3 : Draw_Main_Screen(time);
   else
      close window;

Procedure waitinglist
   Make a vertical Menu
   case currentpos of
      1 : adduserwl;
      2 : deleteuserwl;
      3 : displaywl;
   else
      close window;

Procedure down
   Make a vertical Menu
   case currentpos of
      1 : addrange;
      2 : deleterange;
      3 : list(down);
   else
      close window;

Procedure reserve
   Make a vertical Menu
   case currentpos of
      1 : addrange;
      2 : deleterange;
      3 : list(range);
   else
      close window;
Procedure adduser
    open window & display prompts ('Name', 'SSN', 'Items Checked Out', 'Station Number')
    Display next available station number
    if waiting list size <> 0 then
        display name & ssn from top of waiting list
    getname
    getssn
    getitems
    getnum (get the station number)
    close window

Procedure canceluser
    SelectUserTo(delete)
    open window
    DisplayStudentInfo
    prompt 'Delete this User?'
    getyesno
    if yes then
        close window
        adjust names, times, and station numbers
        Draw_Main_Screen
    if no then
        close window
        go back to Select_User_to

Procedure listuser
    SelectUserTo(list)
    open window
    DisplayStudentInfo
    GetKey
    close window

Procedure switchstations
    SelectUserTo(switch)
    open window
    prompt 'New Station Number: '
    search station numbers for new station number
    if found then
        DisplayStudentInfo (of user selected)
        DisplayStudentInfo (of new station number)
        prompt 'Switch these Users?'
        getyesno
        if yes then
            rearrange station numbers
            Draw_Main_Screen
        if no then
            go back to prompt 'New Station Number:'
    else (if not found)
        rearrange station numbers
    close window
Procedure help
   open window with endtitle 'Press any Key to Continue'
   display information about the function keys and the hot keys
   getkey
   close window

Procedure exit
   open window
   prompt 'Do you wish to leave the system?'
   getyessno (default yes)
   if yes then
      WriteFile(stat)
      Delete Backup Files
   else close window

Function build_statline (receives tempptr)
   statline := day + ',' + month + ',' + year + ',' + hour + ',' +
   minute + ',' + systemname pointed to by system + ',' +
   numinuse pointed to by system + ',' + numwaiting field
   pointed to by system + ',' + loggin + ',' + loggedout

Procedure adduserwl
   open window
   prompt 'Name: '
   getname
   prompt 'SSN: '
   getssn
   add name and ssn to bottom of waiting list
   close window

Procedure deleteuserwl
   SelectUserTo(delete) returns station number
   open window
   write system pointed to by system[i]: station[stanum].student
   write system pointed to by system[i]: station[stanum].ssn
   prompt 'Delete this User?'
   getyessno
   if yes then
      rearrange waiting list
   else
      close window

Procedure displaywl
   open window
   with system do
      for i := 1 to numwaiting
         write waitinglist[i].WL_name;
      getkey
      close window
Procedure adrange (receives statustype)
  open window
  prompt 'Beginning Station Number: '
bEGINNUMBER := getnum
  prompt 'Ending Station Number: '
ENDNUMBER := getnum(default to beginning station number)
  with the current system do
    case statustype of
      down : begin
        NUMDOWN := NUMDOWN + (ENDNUMBER - BEGINNUMBER + 1)
        for i := BEGINNUMBER to ENDNUMBER do
          station[i].status := down
      end
      reserved : begin
        NUMRESERVED := NUMRESERVED + (ENDNUMBER - BEGINNUMBER + 1)
        for i := BEGINNUMBER to ENDNUMBER do
          station[i].status := reserve
      end
    end (case)
  end
  WriteFile(config)
end

close window

Procedure deleterange (receives statustype)
  open window
  prompt 'Beginning Station Number: '
bEGINNUMBER := getnum
  prompt 'Ending Station Number: '
ENDNUMBER := getnum(default to beginning station number)
  with system do
  case statustype of
    down : NUMDOWN := NUMDOWN - (ENDNUMBER - BEGINNUMBER + 1)
    reserved : NUMRESERVED := NUMRESERVED - (ENDNUMBER - BEGINNUMBER + 1)
  end case
  WriteFile(config)
end

close window
Procedure list(receives statustype)
   open window
   with system do
      case statustype of
         down : begin
            write heading line 'Stations Down'
            for i := 1 to numstations do
               if station[i] = down then
                  write i
            end
         reserved : begin
            write heading line 'Stations Reserved'
            for i := 1 to numstations do
               if station[i] = reserved then
                  write i
            end
         end
   close window

Procedure manager
   open window
   prompt 'Enter the Password: '
   getpassword
   if password = pword then
      close window
      managermenu
   else
      open window
      warning('Sorry, that password is Incorrect. ')
      close window

Procedure managermenu
   Make a vertical Menu
   case curntpos of
      1 : addsystem;
      2 : changesystem;
      3 : deletesystem;
      4 : settimes;
      5 : definefiles;
      6 : configurecolors;
      7 : setpassword;
      else
         close window;
   end
   WriteFile(config)
Procedure addSystem
  open window
  prompt 'System Name: '  
  new(newSystem)
  new(newConfig)
  system[numSystems + 1] points to newSystem;
  systemConfig[numSystems + 1] points to newConfig;
  systemname pointed to by system[numSystems + 1] := getName
  prompt 'Beginning Station Number: '  
  beginNumber := getnum (default to 1)
  prompt 'Ending Station Number: '  
  endNumber := getnum (default to 1)
  numSystems := numSystems + 1
  with system configuration at numSystems do
    startingstation := beginNumber
    endingstation := endNumber
    numdown := 0
    numreserved := 0
  end with
  with system[numSystems] do
    for i := beginNumber to endNumber do
      station[i].status := up
      numinuse := 0;
      numavailable := (endNumber - beginNumber) + 1
      numwaiting := 0
    end with

Procedure changeSystem
  open window
  for i := 1 to numSystems do
    write systemname pointed to by system[i]
  SelectSystemTo(change)
  with system configuration do:
    prompt systemname
    getName (default to current value)
    prompt beginNumber
    getnum (default to current value)
    prompt endNumber
    getnum (default to current value)
  close window
Procedure deletesystem
    open window
    for i := 1 to numsystems do
        write systemname pointed to by system[i]
        SelectSystemTo(delete, systemnum)
        if numinuse <> 0 then
            warning('Cannot Delete: There are Still Users on this System')
        else
            prompt 'Are you Sure you want to Delete this System?'
            getyesno
            if yes then
                dispose(system[systemnum])
                dispose(config[systemnum])
                for i := systemnum to (numsystems - 1) do
                    system[i] pointer := system[i + 1] pointer
                    config[i] pointer := config[i + 1] pointer
                close window

Procedure settimes
    open window
    prompt 'Minutes between backups: [ ]'
    backuptime := getnum(default to current value)
    prompt 'Save Statistics every (Minutes): [ ]'
    stattime := getnum(default to current value)
    close window

Procedure definefiles
    open window
    prompt 'Full pathname for Backup File (drive & directory): '
    backuppath := getpath(default to current path)
    prompt 'Full pathname for Statistics File (drive & directory): '
    statpath := getpath(default to current path)
    prompt 'Full pathname for Message of Day File (drive & directory): '
    MODpath := getpath(default to current path)
    close window

Procedure configurecolors
    open window
    repeat
        display the items to be configured and the colors available
        use point and shoot to select the item and color
        store value in colorconfig
        display the item and color selected in a small window
    until item = exit
    close window
Procedure setpassword
open window
prompt 'Enter the old password: '
old := getpassword
prompt 'Enter the new password: '
new := getpassword
prompt 'Re-enter the new password: '
new2 := getpassword
if (old = pword) AND (new = new2) then
  pword := new
else
  warning('Sorry, that password is incorrect')
GetKey
close window

Function getpassword(returns a string password)
acceptablechars := ['A'..'Z','a'..'z','1'..'9']
maxlength := 5
getpassword := getstr(pas the row,col,maxlength,
  acceptablechars, initvalue,color,acceptableterm,echo)

Function getpath(returns a string pathname)
acceptablechars := ['A'..'Z','a'..'z','1'..'9',';','\','.']
maxlength := 255
getpath := getstr(pas the row,col,maxlength,acceptablechars,
  initvalue,color,acceptableterm,echo)

Function getyesno(returns 'Y' or 'N')
acceptablechars := ['Y','y','N','n']
maxlength := 1
getyesno := getstr(pas the row,col,maxlength,acceptablechars,
  initvalue,color,acceptableterm,echo)

Function getstr(receives row,col,maxlength,acceptablechars, initvalue,color,acceptableterm,echo; returns term, str)
repeat
  GetKey(return ch, term)
case term of
    kascii : begin
      if (ch IN acceptablechars) then
        if echo
          write ch
        else
          write '#'
      end
    until term in acceptableterm

Function getname(returns the name string)
acceptablechars := ['A'..'Z','a'..'z']
maxlength := 20
getname := getstr(pas the row,col,maxlength,acceptablechars,
  initvalue,color,acceptableterm,echo)
Function `getitems` (returns the item string)
   acceptablechars := ['A'..'Z','a'..'z','0'..'9',',' ]
   maxlen := 40
   getitems := getstr(pas the row, col, maxlen, acceptablechars, inivalue, color, acceptableterm, echo)

Function `getssn` (returns the ssn string)
   acceptablechars := ['0'..'9', '-' ]
   maxlen := 11
   getssn := getstr(pas the row, col, maxlen, acceptablechars, inivalue, color, acceptableterm, echo)

Function `getnum` (returns the numeric string)
   acceptablechars := ['0'..'9', ' ' ]
   getnum := getstr(pas the row, col, maxlen, acceptablechars, inivalue, color, acceptableterm, echo)

Function `selectuserto` (receives actiontype)
   \( r := 1 \)
   \( c := 1 \)
   repeat
      sprint(pas r,c,strdat,color = 1)
      getkey(ch,term);
      sprint(pas r,c,strdat,color = 0)
      case term of
         fup : if (r > 1) then r := r - 1
         fdown : if (r < maxrow) then r := r + 1
         fright : if (c < maxcol) then c := c + 1
         fleft : if (c > 1) then c := c - 1
      end
   until term IN [fesc,freturn]

Function `selectsystemto` (receives actiontype, returns systemname)
   repeat
      set r = initial row
      set c = initial column
      print window(pas r,c,strdat,color = 1)
      getkey(ch,term)
      print window(pas r,c,strdat,,color = 0)
      case term of
         fup : (if r > 1) then r := r - 1
         fdown : (if r < maxrow) then r := r + 1
         fright : (if c < maxcol) then c := c + 1
         fleft : (if c > 1) then c := c - 1
      end
   until term IN [fesc,freturn]
Procedure systems
    open window
    i := 1
    write systemname pointed to by system[i]
    for i := 1 to numsystems do
        write systemname pointed to by system[i]
    end while
selectsystemto(change) returns name
    i := 1
while systemname pointed to by system[i] <> systemname returned
    i := i + 1
    current := i
close window

Procedure Draw_Closing_Screen
    open window
    write 'Thank You for using the Computer Laboratory Monitoring System'
    getkey
    close window

Procedure escape
    close window

Procedure Warning(receives msg)
    open window with endtitle 'Press any Key to Continue'
    write msg
    getkey
    close window