A "UNIX" BASED
ELECTRONIC CALENDAR SYSTEM

by

DAVID OWEN JAMES

B. A., Bethany College, Lindsborg, Kansas, 1981

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1982

Approved by:

Major Professor

ACKNOWLEDGMENT

I wish to thank Dr. Richard A. McBride for his guidance and encouragement throughout the course of this project.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.0 Overview

Appointment calendars are time management tools that allocate specific time segments for activities on a given day. Electronic calendars are computerized versions of the more traditional paper counterparts, but with several advantages, including a wider variety of useful applications. This report contains a description of an implementation of such a system in Pascal on the UNIX operating system currently running on a 32-bit Perkin-Elmer 3220 at the KSU Department of Computer Science.

1.1 Rationale of Electronic Calendars

The obvious advantages of an automated calendar system are the same benefits derived from automating other office processes and transactions: fast access time of computer stored information, storage space efficiency, and automatic (periodic) transaction processing. But many other advantages of automation of appointment schedules exist.

Studies by Mitzberg (1971), and also Boin (1978) have shown that managers and other professionals spend 40-70 percent of the working day in meetings. Costly problems resulting from missed meetings or late arrivals often arise from the fact that individual appointment schedules fall

behind in currency, or that conflicts are inadvertently scheduled during meeting times. If the appointment schedules are manually maintained, random appointment scheduling, such as weekday appointments that don't vary from week to week, can become error prone. Automation in this area could solve many of these problems.

Another advantage of electronic calendar systems can be attributed to appointment scheduling between several principals or users. "Secretaries find scheduling meetings one of the most distasteful aspects of their job, involving many frustrations." (Bancomb, 1981) An Appointment to be made involving several people must be in the intersection of each member's free time schedule, so to speak, but that intersection is not known all at once. "Typically the initial scheduling is done in several passes starting with collection of scheduling constraint information followed by negotiation and selection of a meeting times." (Greif, 1982) A particular time slot that is acceptable to, say, the first three members may not be possible for the fourth. Thus, the process has to be started over, with another attempted proposed meeting time. A study by Bacomb (1981) of this process showed that an average meeting of six people took 60-75 minutes to schedule. In this application, automation is clearly worthwhile. If principal's schedules were kept on computer files instead of disjoint pieces of paper, the "free time intersection" for any number of participants

could easily be calculated. From that information, appointments could easily be set.

1.2 Scope of Implementation

The advantage of electronic calendars systems as cited in the previous sections can be immediently applied to scheduling problems in the KSU Department of Computer Science. Each faculty member keeps some sort of appointment calendar (possibly implicit), to record appointments. Many appointments, such as classes being taught or taken, recur weekly, that is, they are generally the same from week to week. Since the faculty's schedules are often (purposely) staggered, it is often particularly difficult to schedule appointments among them. The automated scheduling algorithms of an electronic calendar system could be a valuable assistance to the administration and students of the department.

This particular implementation of an electronic calendar is designed for use under the UNIX operating system at the KSU Computer Science department. The decision for that choice derives from the fact that UNIX is an industry - wide de facto standard for operating systems on minicomputers, and also current availability of computer resources. There are interactive terminals interfaced to the UNIX system currently operating in most of the offices in the department, so the primary users, the KSU Computer

Science faculty, will have convenient access to the electronic calendar system. A principal, or calendar user, in this application will be defined as any faculty member or graduate student who has possession of a valid UNIX account. Also, a principal could be a non-person, such as a room or other resource, for which daily appointment schedules can be automatically maintained.

The implementation tool is Pascal, specifically, UNIX Berkeley Pascal. (5-7) This choice is, again, primarily based on availability. But it also is a result of the high extent of use of the Pascal language in the department, and the fact that the language and operating system are fairly standard, for portability concerns.

CHAPTER 2

PROBLEM STATEMENT

2.0 Introduction

An electronic calendar system should be written for people to use and benefit from, as a cooperative activity. (Greif, 1982) A priority is ease and convenience in typical daily use. A primary requirement is that the electronic calendar be made available for use interactively 24 hours a day in a multi-user environment. Since all steps performed by the electronic calendar are transmitted to and from the user via interactive CRTs, the level of information displayed to the user should be user-friendly, but concise enough to insure speed and effiency in use by more experienced users.

2.1 Requirements

Most often, invocation of a calendar program will lead to executing one or two commands, such as looking at tomorrow's schedule, or requesting a meeting. Therefore, the number of steps required for the user to execute the most common functions be minimal in number, so that the steps can be executed as quickly and as easily as possible.

The more specific requirements related to actual features of a running electronic calendar system can be divided into two major categories. The first is maintenence

of a principal's personal calendar. Maintenence includes insertions and deletions of scheduled appointments. These functions should be applicable to any particular day desired within six months of the current day. A related required feature is that, at the user's option, the insertion or deletion of an appointment should be allowed to apply globally to all subsequent weeks, on the same weekday. As an example, a change for, say, Wednesday, the third of the month, could either only apply to that particular day, or else apply to that particular day and all subsequent Wednesdays, also. In the latter case, the dates affected would be 3, 10, 17, etc. This will be useful for recurring weekly appointments that generally don't change from week to week.

The second category of requirements applies to use of an electronic calendar system in scheduling meetings between two or more principals. A mechanism that allows an appointment request to be sent to all involved principals should be made available to all users of the calendar system. The desire is to only assume confirmation of a requested meeting when all the principals involved have agreed to the request. Each person to whom a request is sent should be notified that the outstanding request does exist, and that an answer is expected. If all requests are answered in the affirmative, the meeting is considered confirmed. If any of the principals rejects the requested

meeting, the meeting is considered canceled, and the members notified, so that possibly another attempt can be made to schedule a meeting.

To both of the previously described categories, some specific requirements apply. Under no circumstances will time conflicts be allowed, that is, an electronic calendar system should always prevent any step executed which would result in overlapping appointments, such as appointment "A" at 8:00 AM - 8:30 AM on the same day as appointment "B" at 8:15 AM - 8:45 AM. Provisions for notifying the user attempting the offending insertion should be made, the particular action to be taken being dependent on the context of the state.

The per-day clock should be one of length 24 hours. Thus, a 2:00 AM - 3:00 AM appointment would be possible. However, normal operation of the calendar should assume an 8:00 AM - 5:00 PM work day, with that default possibly being overridden by the user.

As a final requirement for an electronic calendar system, some provision for backup and purging of old calendar information must be made. The information about past dates does not necessarily need to be directly accessible to the user, but some means for outdated information retrieval must exist.

2.2 Specifications

This section will contain a refinement of the previously defined set of requirements for an electronic calendar system. The refinement is a set of specifications, which precisely defines the outcome of any specific solution to the problem. These specifications do not imply design of the solution, but merely the goals which any answer to the problem must meet.

The electronic calendar system is menu driven; that is, it follows a prompt - and - response format. Clear, concise prompts that require very short responses of the user will be implemented, to assure the speed and efficiency of each execution step at runtime.

In development of the prompts organization and ordering, the list of all possible desired steps executed can be observed. In categorizing the steps, one possibility follows.

1. Observe or edit a particular day's schedule.
2. Observe or edit the events of a typical week-day's schedule which will apply to all succeeding weeks.
3. Send a request to any number of users for a meeting for a particular day.
4. Answer requests made by other users to attend a meeting for a particular day.

While there are clearly other categorizations of steps in an electronic calendar system, such as combining steps one and

two, and/or combining steps three and four, the point to be made here is that each category chosen will entail a different set of prompts, at least to some extent. Therefore a menu-driven format appears to lend itself to this application well. The size of each menu and the number of menus (which are inversely proportional) are determined such that, again, convenience for the user is maximized.

The description of prompts can be most easily conveyed in outline form. Each entry in the following outline will represent a set of one or more prompts required for the transaction to be carried out.

A. Observe or Edit Personal Calendar.
    1. For a Particular Day.
        a. Observe.
        b. Insert.
        c. Delete.
    2. For a Typical Weekday.
        a. Observe.
        b. Insert.
        c. Delete.

B. Make or Send a Meeting Request.
    1. Make a Request.
    2. Send a Request.

C. Quit.

Two other requirements also related to prompts are "user friendliness", and convenience. Some specifications

consistent with these requirements can be deduced. The software must be robust. It should accept a wide variety of input as valid answers to prompts. A typical example of this is, as follows: After a prompt for a time of day, either "8:00 AM", "8:00AM", "8:00", or "800" will be accepted. If the program cannot interpret an answer, it will reprompt with a helpful error message, and, of course, the program will never crash as a result of invalid input.

In addition to syntatic input checks, all semantic checks will be performed to be consistent with the "no time conflicts" requirement. This applies any time an insertion attempt is made to any existing schedule.

The electronic calendar system will only make requests for meetings that are scheduled during the "free time intersection" of all principals, in order to prevent time conflicts. However, the option to reject a meeting (presumably for some other reason) is available to any of the principals. In the course of making and answering requests, a more direct communication between users is required than the ones previously described. Therefore, a link to the computer system's interactive mail service will be created to insure currency of users on meeting requests and responses. This process will fulfill the requirements of member notification in scheduling meetings.

A function to backup and purge computer stored information about appointments for past dates will be

provided. The software for this function will be either manually or automatically (periodically) invoked. The details of that process will depend on the current status of available resources of storage devices.

CHAPTER 3

SAMPLE SCENARIO

3.0 Introduction

One way to better understand the implementation of any software product is to observe the operation and results of the program(s) in actual use. For a highly interactive program, such as an electronic calendar, for which prompts - and - responses make up the bulk of its operation, an annotated scenario is of value.

This chapter contains such a scenario which applies specifically to the writer's implementation of an electronic calendar system at the KSU Department of Computer Science. The chapter is intended as a general user's manual, for those users of the UNIX system who desire the facilities provided by the programs of the electronic calendar. However, the objectives implied by this scenario can be applied to electronic calendars in general. It describes a solution to the problem of meeting the formerly defined specifications for any electronic calendar system.

In illustration of the prompts, and the possible responses to those prompts, not all possibliities of input errors will be included in the scope of the chapter. The software is written such that a wide variety of answers to prompts is accepted. This philosophy is consistent with the "user-friendly" requirement. As a specific example; upper

and lower case letters are generally not differentiated - when answering a prompt an alphabetic character may be upper or lower case. The number of "loop until good data" constructs has been kept to a minimum, but there are instances where an escape from a prompt requesting input does not exist. Error messages are helpful in pinning down explanations as to why data is bad. If abnormal termination is desired at any point in the program, a break in the program execution (through the "break" key on most terminals) will not in any way cause harm to the calendar data files used by the program.

The description and illustrations of the sample scenario can be divided into two categories: functions generally dealing with personal appointments, in which the user is the primary principal, and meetings, where possibly several different users of the system share a common calendar entry for a specific date and time. This division is shown in the main menu. As an example, the display seen at invocation of the electronic calendar is:

```
Welcome to Electronic Calendar

Today is Thu, Nov 18.

Main Menu:
1.  Update or Observe Personal Calendar.
2.  Make or Answer Meeting Requests.
3.  Exit.

Please enter command number or X. ->
```

The description of the processes contained in choice (1) is discussed in section 3.1, and the processes of choice (2)

are described in section 3.2.

3.1 Personal Calendar Operations

Choosing the first option from the initial main menu puts the user in a mode intended for formatted display and editing of the files contained in that user's account that make up his/her personal calendar. It is assumed that this mode would be used fairly routinely, but for short periods of time at each setting. Therefore the prompts as described here reflect a logical consistency with the order in which decisions would have to be made to observe or edit a manual calendar. The first decision that will have to be made by the user is the date that he/she wants to look at, and possibly edit. The prompt is:

<Ret> for todays schedule, enter day or date. ->

The answer to this prompt is of some importance, in that the scope of the resulting schedule and any edits to that schedule is determined by the user's response. Four possibilities for answers exist: (1) a null line <return>, (2) a date (eg. '24'), (3) a month and date (eg. 'Nov18'), and (4) a weekday (eg. 'Thu'). The scope differences are these. Choices (1) through (3) yield a specific day, in which any changes made only apply to that day. Choice (4) yields a weekday schedule, in which any changes made apply to all succeeding days in the entire calendar which are of the same weekday.

The differences between choices (1), (2), and (3) are minor, the differentiation is primarily a matter of convenience. With the assumption that the most common day to be observed is the current day, a null line entered; that is, an enter (return) key depressed with no previous characters entered, yields the current day's schedule. Choice (2) is equivalent to choice (3), except that the month can only by ommitted (as in choice (2)) if the date is within two weeks of the current day.

An example will help illustrate. Suppose that on November 18, 1982 a user wants to observe or edit his personal calendar. Invocation of the electronic calendar system displays the following:

```
Welcome to Electronic Calendar.

Today is Thu, Nov 18, 1982.

Main Menu:
1. Update or Observe Personal Calendar.
2. Send or Receive Meeting Requests.
X. Exit.

<Ret> for todays schedule, enter day or date. ->
```

At this point, to observe or edit the current day's schedule (November 18), the user could enter a null line, '18', 'Nov18', or one of several variations of 'Nov18' to yield exactly the same results. The schedule for November 18 will be be displayed. If the schedule for, say, Friday, November 19 is to be observed, either '19' or 'Nov19' (or variations) could be entered, with equivalent results. In this example, a single date, without the corresponding month, could be

entered for dates up to and including Wednesday, December 1, which is the two week limit for assumed month names. For any desired date past December 1 (up to a year from the current day), both the month and date must both be given.

To observe a weekday's schedule; that is, the appointments that generally apply to all weeks, the name of the weekday is entered. A three letter abreviation (eg. 'Thu') is expected, but several variations are accepted. Any changes performed on a weekday schedule will apply to all succeeding weeks.

After successful completion of the date selection process, the schedule for the date (or weekday) is displayed. A heading for the display indicates either the specified day and month, or the general weekday, for which the following schedule applies. The actual schedule follows, and then the list of options which apply to that schedule. A sample schedule for a user named Rich follows:

```
Schedule for Thu, Nov 18:

8:00 - 9:00     CS420 Operating Systems (teaching)

10:30 - 11:45   CS960 Theory of Database

12:00 - 1:00    Lunch

3:00  - 3:30    virg  rich  beth  rod
Confirmed by:   virg  rich  beth
                To Discuss Curriculum Changes.
                Meet in F112.

4:30  - 5:00 R1 virg rich
                to discuss UNIX-OS/32 Networking.

(I)nsert, (D)elete, (L)ist, (N)ewday, or e(X)it->
```

Some discussion of the displayed sample schedule is warranted. The first three appointments are personal appointments that generally only apply to Rich's schedule. The 3:00 - 3:30 meeting is with four members, including Rich. Members Virg, Rich, and Beth are confirmed members; that is, they have all affirmatively answered a request for the meeting. Rod has not answered the request either way, from the fact that his name is in the "members" set, but not in the "confirmed" set. The "confirmed by" field in an indication that the meeting was originally requested by Rich, so that he is the owner of that meeting. The "confirmed by" field does not appear for that meeting on the other member's schedules for that day. 5:00 appointment shows up as a request, indicated by the "R" field. This means that Rich still needs to answer the request for that appointment.

A discussion of making and answering requests is left to the next section of this chapter. They are briefly mentioned here, as the entries which contain members are special cases for the delete routines (to be discussed).

The choices of input can now be dealt with. The first possibility calls for an insert process to be invoked. This process logically requires two pieces of information from the user, the time span of the appointment and a description of the appointment. Note that all appointments involving more that one user (refered to as "meetings") are not

inserted here, but through the seperate request and answer procedure. After an 'I' is entered as an answer to the last prompt, the time span for the appointment to be inserted is asked for:

Enter Beginning and Ending Times. ->

after which the times can be entered. The formats in which this information is entered can vary. For instance, colons (":") may be either included or ommitted, as can "AM" or "PM". In the latter case, certain assumptions are made. In particular, 1:00 - 6:55 times are assumed "PM", and 7:00 - 12:55 times are assumed "AM". Thus, in order to schedule an appointment at a time outside of these defaults, the "AM" or "PM" must be included, such as for 7:30 PM, or 6:00 AM.

The electronic calendar system will not allow conflicts. If the time entered in any way conflicts with any previous appointment of that day, the user is informed of the conflict and returned to the previous prompt. After a non-conflicting time span is entered, a prompt for the appointment description is displayed. An example of an insertion of Rich's schedule follows, as a continuation of the previously desplayed sequence. The first insert attempt is a conflict.

```
(I)nsert, (D)elete, (L)ist, (N)ewday, or e(X)it -> I
Enter Beginning and Ending Times. -> 11:30 12:00
11:30 - 12:00 conflicts with a previous
              10:30 - 11:45 appointment.
(I)nsert, (D)elete, (L)ist, (N)ewday, or e(X)it -> I
Enter Beginning and Ending Times. -> 11:45 12:00
Enter Description of appointment.
```

At this point a description can be entered. The description can be up to 80 characters, and is terminated by a carriage return.

The insert process just described can also be applied the same way to a general weekday's schedule. The only difference is that the insertion applies to more than just the one day's schedule. A general weekday schedule is indicated by the heading. For each insertion the entered time span is the compared with to flag conflicts. Upon completion of entering the description, an attempt is made to insert the new appointment to all days present in the calendar system of the same weekday. The results of the attempt are then displayed.

To illustrate, if the above insertion was made to a general Thursday schedule instead of the specific Thursday, November 18 schedule, the response after the appointment description was entered might have been:

```
11:45 - 12:00 appointment inserted to Thu, Nov 18.
11:45 - 12:00 appointment inserted to Thu, Nov 25.
Cannot insert to Thu, Dec 1, due to previous
              11:30 - 12:00 appointment.
Insertion made to general Thu schedule.
```

If the appointment is desired for December 1, it would have to be manually inserted, after a deletion of the conflicting appointment of the same day. The dates for which the trace message are displayed are limited to those dates which have appointments scheduled other than weekly appointments. The change made implicitly applies to all succeeding Thursdays

(December 8, 13, ...).

At this point the insertion process is concluded, and the higher level prompt is displayed:

(I)nsert, (D)elete, (L)ist, (N)ewday, or e(X)it->

The next possibility for an answer to this prompt is delete, which behaves similiarly to insert, but is even simpler to use. All that is required by the delete process is to know which appointment to delete. The easiest way to uniquely identify an appointment is by its beginning times. The prompt is:

Enter beginning time of appointment to delete. ->

The expected answer to this prompt should follow the same format and defaults at the prompt for beginning and ending times described earlier for the insert process, except, of course, that only one time need be entered. The case where the time entered does not exist as the beginning time of an appointment is indicated by a prompt, such as:

Appointment with beginning time 11:45 not found.
(I)nsert, (D)elete, (L)ist, (N)ewday, or e(X)it ->

and control is then returned to the previous prompt.

From the user's point of view, the one entry for a beginning time is all that is required. But the effects resulting from special cases of deletes need to be defined.

If the schedule from which the appointment was deleted was a weekday schedule, then the delete is also attempted on all successive dates having the same weekday. Similiar trace messages are displayed. As an example:

```
11:45 - 12:00 appointment deleted from Thu, Nov 18.
11:45 - 12:00 appointment deleted from Thu, Nov 25.
11:45 - 12:00 appointment not found on Thu, Dec 1.
11:45 - 12:00 appointment deleted from Thu schedule.
```

Another case of delete is when the appointment to be deleted is a meeting of several users; that is, the appointment entry contains a list a participants. If such a meeting is deleted, then one of two events occur. If the user performing the deletion is the owner of the meeting; that is, the one that requested the meeting in the first place, then the meeting is considered cancelled. It is automatically deleted from all of the member's schedules, and mail is sent to each member informing him/her of the cancellation. If, instead, the user performing the deletion is a member of the meeting, but not the owner, then the deletion is only made to his/her schedule, but his name is automatically removed from the list of members on all of the member's schedules. Also, mail is sent to the owner of the meeting informing him/her of the ommission of the one member from the meeting.

Three other possibilities for answers to the second level prompt exist, other that insert or delete. The "List" and "Newday" options perform backtracking. The "List" re-displays the day's schedule and the same prompt, and the "Newday" option displays the prompt for date selection (discussed previously), so that a different date/day can be observed or edited. The "Exit" choice returns control to the initial main menu, at the next higher level.

3.2 Meeting Requests

The second choice available at the main menu level is the "Make or Answer Requests" selection. The processes included in this subdivision are so categorized from the fact that all the calendar entries dealt with by this division are meetings, all including more than one member and an "owner" of the meeting. As previously described in the requirements / specifications, a meeting is first requested by one user to several members. When the requests are all answered affirmatively, the meeting in considered confirmed.

The first prompt after selection of this subdivision from the main menu is:

(R)equest a meeting, or (A)nswer requests? ->

If an "R" is entered, the request process is invoked. This process is similiar to the insertion process previously described, in that a date, time, and description must be entered by the user, and that the appointment is inserted to the user's schedule for the particular data. The set of prompts making up the insertion process is augmented with additions allowing the insertion to apply to several members, as requests. The next prompt observed for the request process is:

With whom would you like an appointment?
Enter names or list. ->

Here the names entered must match the login names known by the system under which the electronic calendar is

implemented. The "list" option is often useful when the names are not absolutely known; the names of all users participating in the electronic calendar system (as they must be entered) will be listed, and the prompt will be re - displayed. An example with four total users is:

```
With whom would you like an appointment?
Enter names or list. -> list
beth   virg   rich   rod
With whom would you like an appointment?
Enter names or list. -> beth virg
For how long? ->
```

The value entered for the duration of the proposed meeting must be a multiple of 5 (5, 10, 15, ...). The next prompt asks for the date or month and date of the proposed meeting:

```
Enter Preferred date, or month and date. ->
```

The response to this prompt follows the same guidelines as the similiar prompt at the beginning of the "Update or Observe" process; a null line implies the current date, a single date can be entered for dates up to two weeks from the current date, and past that date the month name is required. The only exception is that weekday names are not allowed. The next prompt is:

```
Standard 8:00 - 5:00? (Y/N) ->
```

A "yes" answer to this prompt will limit meeting possibilities to the time spans between 8:00 AM and 5:00 PM. Otherwise, all of the possibilities during the 24 hour day will be listed.

At the successful completion of a date selection and the decision of the "Standard?" question, the "free time

intersection" of all the members entered and the user are calculated and displayed. An example of the display is:

Here are the free times for Thu, Nov18:

9:00 - 10:00 2:00 - 2:30

Do you want one of these? Y/N/e(X)it ->

The time slots listed represent the free time intersection of all the members (including the owner) for that particular day. Only the time pairs that represent durations greater than or equal to the previously entered time span are displayed.

The associated prompt asks for one of three choices. A "yes" answer to the prompt means that the user does wish to proceed with the requesting process, on the date to which the displayed possibilities apply. In that case the prompt for beginning and ending times of the requested meeting appears:

Enter Beginning and Ending Times. ->

The format for the expected input is exactly the same as the time pairs expected by the insertion process, described in the previous section. The time pair entered is similiarly validated by checking against the formerly displayed times for conflicts. After a conflict - free time pair is entered, a prompt for the meeting description is displayed, again analogous to the insertion process:

Enter description of meeting. ->

When the description is entered the request process is