

https://dspace.org/

Design of a disk-based file server involving the mapping of "pseudo-disks"

Hildyard, Mark William

2015-01-27
http://hdl.handle.net/10539/16783

Downloaded from a DSpace repository. https://dspace.org/

```
            Status = NOT (Ready);
            DO WHILE (Status <> Ready);
              CALL Op$Table_Disks$Ready (First$D$Num, Qty, Status$Ptr);
              IF (Status <> Ready)
                THEN
                  DO;
                    /* Allow other tasks to continue */
                    CALL RQ$SEND$CONTROL (@Exception);  /* Release Op$Table Resource */
                    CALL RQ$SLEEP (Time$Len, @Exception);
                    CALL RQ$RECEIVE$CONTROL (Op$Table$Region, @Exception);
                  END;
            END; /* While - Files are open for use */
          END;
      END Mark$Open$And$Wait;


/*PROCEDURE */ Mark$All$Open: Procedure;
    /* Mark all Disks of the application as open - Files are already open */

    /*BEGIN*/
      IF ( NOT(Reqd$Appl.All$Reqd) )
        THEN Status = Cannot$Load$All;
        ELSE
          DO;
            Qty = Reqd$Appl.Qty;  /* Set the output for the number of disks */
            CALL Op$Table_Search$Disk (Rqst$Disk, First$D$Num$Ptr, Status$Ptr);
            CALL Mark$Open$And$Wait (Qty);
          END;
      CALL RQ$SEND$CONTROL (@Exception);  /* Release Op$Table Resource */
    END Mark$All$Open;


/*PROCEDURE */ Mark$Single$Open: Procedure;

    /* Mark Reqd Disk Open in the Op$Table - Attach first & open file if necessary */

    /*BEGIN*/
      /* Set the output for the number of disks */
      Qty = 1;
      /* Ensure that the disk does exist */
      IF ( (Rqst$Disk - Rqst$Appl) > Reqd$Appl.Qty)
        THEN
          DO;
            Status = Disk$Not$Exist;
            CALL RQ$SEND$CONTROL (@Exception); /* Release Op$Table */
          END;

        ELSE /* Disk does exist */
          DO;
            CALL Op$Table_Search$Disk (Rqst$Disk, First$D$Num$Ptr, Status$Ptr);
            IF (Status = Found)
              THEN  /* Disk has already been used */
                DO;
                  CALL Mark$Open$And$Wait (1);
                  CALL RQ$SEND$CONTROL (@Exception);  /* Release Op$Table Resource */
                END;

              ELSE  /* Disk has not yet been used */
                DO;
                  /* Create Entry in Op$Table Resource & Mark it as open */
```

```
                    CALL Opt$Table_Attach$Disks (Rqst$Disk-1,Opt$Appl$Num,First$D$Num$Ptr,Status$Ptr);
                    IF (Status = Successful)
                      THEN
                        DO;
                          CALL Opt$Table_Mark$Open (First$D$Num, 1, Read$Write, Status$Ptr);
                          IF (Status <> Successful)
                            THEN  /* Restore Opt$Table */
                              CALL Opt$Table_Remove$Disks (First$D$Num, 1);
                        END;
                    CALL RQ$SEND$CONTROL (@Exception);  /* Release Opt$Table Resource */

                    /* Open the files on the Hard Disk */
                    IF (Status = Successful)
                      THEN
                        DO;
                          /* Attach & Open files on the Hard Disk */
                          CALL RQ$RECEIVE$CONTROL (HD$Attach$Region, @Exception);
                          CALL HD$Attach_Open$Create$File ((Rqst$Disk MOD 16),Reqd$Appl.Appl$Connection,
                                Read$Write, @Opt$Disks$Map(First$D$Num).Open$Connection, @Exception);
                          IF (Exception <> E$OK)
                            THEN Status = Disk$Attach$Error;
                          CALL RQ$SEND$CONTROL (@Exception); /* Release HD$Attach Resource */

                          /* If an error occured then restore the Opt$Table Resource */
                          IF (Status <> Successful)
                            THEN
                              DO;
                                CALL RQ$RECEIVE$CONTROL (Opt$Table$Region, @Exception);
                                  CALL Opt$Table_Mark$Closed (First$D$Num);
                                CALL RQ$SEND$CONTROL (@Exception);
                              END;
                        END;
                  END; /* Disk Entry Created */
              END; /* Disk required is a valid disk */
        END Mark$Single$Open;



/*PROCEDURE */ Get$New$Appl: Procedure;

    /* Reads the application's parameters from the Common$Info Resource, Determines
        access & obtains a token for the application's directory file */

    /*BEGIN*/
        CALL Common$Info_Read (Rqst$Appl, Status$Ptr);
        IF (Status = Successful) THEN
          IF ( ( (Rqst$Disk - Rqst$Appl) > Reqd$Appl.Qty) AND (NOT Whole$Appl$Rqst) )
            THEN Status = Disk$Not$Exist;
            ELSE
              DO;
                CALL Query$Access (@New$Appl.Access, New$Appl.Load$Type, New$Appl.OS$Num);
                IF (Status = Granted)
                  THEN  /* Can attach the Applications Directory */
                    DO;
                      CALL Common$Info_Attach (Status$Ptr);
                      IF (Status <> Successful)
                        THEN CALL RQ$A$DELETE$CONNECTION (New$Appl.Appl$Connection,
                                                          No$Mailbox, @Exception);
                    END;
              END; /* Access determined & directory attached */
```

```
    END Get$New$Appl;


/*PROCEDURE */ Load$All$Disks: Procedure;

    /* Each disk of the new application is attached to the Op$Table.  The required
       disk is marked as open & the files for the disks are opened */

    /*BEGIN*/
      /* Set the output for the number of disks */
      IF (Whole$Appl$Rqst)
        THEN Qty = Reqd$Appl.Qty;
        ELSE Qty = 1;

      CALL Op$Table_Attach$Disks (Rqst$Disk, Reqd$Appl.Qty, Op$Appl$Num, First$D$Nums$Ptr, Status$Ptr);
      IF (Status = Successful)
        THEN
          DO;
            /* Mark all disks of the application or just the required disk as open */
            IF (Whole$Appl$Rqst)
              THEN CALL Op$Table_Mark$Open (First$D$Num, Reqd$Appl.Qty, Read$Write, Status$Ptr);
              ELSE CALL Op$Table_Mark$Open ( (First$D$Num + (Rqst$Disk MOD 16) - 1), 1,
                                                    Read$Write, Status$Ptr);


            IF (Status <> Successful)
              THEN  /* Restore Op$Table */
                CALL Op$Table_Remove$Disks (First$D$Num, Reqd$Appl.Qty);
              ELSE
                DO;
                  /* Open or create files on the Hard Disk */
                  CALL RQ$SEND$CONTROL (@Exception);   /* Release Op$Table Resource */
                  CALL RQ$SEND$CONTROL (@Exception);   /* Release Common$Info Resource */
                  CALL RQ$RECEIVE$CONTROL (HD$Attach$Region, @Exception);
                  DO Count = 0 TO (Reqd$Appl.Qty - 1);
                    CALL HD$Attach_Open$Create$File ( (Rqst$Disk + Count), Reqd$Appl.Appl$Connection,
                      Read$Write, @Op$Disk$Map(First$D$Num + Count).Open$Connection, @Exception);
                  END; /* Do Loop */
                  CALL RQ$SEND$CONTROL (@Exception);   /* Release HD$Attach Resource */
                END;
          END; /* Else */
        IF (Status <> Successful)
        THEN /* Restore Op$Table */
          CALL Op$Table_Remove$Appl (Op$Appl$Num);
    END Load$All$Disks;



/*PROCEDURE */ Load$Single$Disk: Procedure;

    /* Attach the disk to the Op$Table (create an entry), mark as open &
       open the file associated with the disk */

    /*BEGIN*/
      /* Set the output for the number of disks */
      Qty = 1;

      CALL Op$Table_Attach$Disks (Rqst$Disk, 1, Op$Appl$Num, First$D$Num$Ptr, Status$Ptr);
      IF (Status = Successful)
        THEN
          DO;
```

```
            /* Mark disk entry as open */
            CALL Op$Table_Mark$Open (First$D$Num, 1, Read$Write, Status$Ptr);
            IF (Status <> Successful)
              THEN  /* Restore Op$Table */
                CALL Op$Table_Remove$Disks (First$D$Num, 1);
              ELSE
                DO;
                  /* Open or create file on the Hard Disk */
                  CALL RQ$SEND$CONTROL (@Exception);   /* Release Op$Table Resource */
                  CALL RQ$SEND$CONTROL (@Exception);   /* Release Common$Info Resource */
                  CALL RQ$RECEIVE$CONTROL (HD$Attach$Region, @Exception);
                  CALL HD$Attach_Open$Create$File ( (Rqst$Disk MOD 16), Reqd$Appl.Appl$Connection,
                        Read$Write, @Op$Disk$Map(First$D$Num).Open$Connection, @Exception);
                  CALL RQ$SEND$CONTROL (@Exception);   /* Release HD$Attach Resource */
                END;
            END; /* Then */
        IF (Status <> Successful)
          THEN  /* Restore Op$Table */
            CALL Op$Table_Remove$Appl (Op$Appl$Num);
      END Load$Single$Disk;



/*BEGIN  Common$Disk$Open */

    /* If Rqst$Disk is an application number then all disks of the application are required,
         else only the single disk requested is required */
    Rqst$Appl :  (Rqst$Disk AND 0Fh);  /* Drop the last 4 bits */
    IF (Rqst$Appl = Non$Appl)
      THEN Status = Appl$Not$Exist;   /* Nothing further will be attempted */
      ELSE


  DO;  /* Rqst$Appl <> Non$Appl */

    IF (Rqst$Disk = Rqst$Appl)
      THEN
        DO;
          Whole$Appl$Rqst = True;
          Rqst$Disk = Rqst$Appl + 1;  /* First Disk of Appl */
        END;

   RESTART:

    /* Is the Application already in the Op$Table */
    CALL RQ$RECEIVE$CONTROL (Op$Table$Region, @Exception);
    CALL Op$Table_Search$Appl ( Rqst$Appl, @Op$Appl$Num, Status$Ptr);
    IF (Status = Found)

      THEN  /* Case Where Application is already in use, and some files available. */
        DO;
          /* Set up pointer to the relevant application entry */
          Reqd$Op$Appl$Ptr = @Op$Appl$Map(Op$Appl$Num);
          CALL Query$Access (@Reqd$Appl.Access, Reqd$Appl.Load$Type, Reqd$Appl.OS$Num);
          IF (Status <> Granted)
            THEN CALL RQ$SEND$CONTROL (@Exception);  /* Release Op$Table Resource */
            ELSE IF (Whole$Appl$Rqst)
                   THEN CALL Mark$All$Open;
                   ELSE CALL Mark$Single$Open;
        END; /* Case where Application is already available */
```

```
            ELSE  /* Application is not yet in use - Must load */
              DO;
                CALL RQ$SEND$CONTROL (@Exception);  /* Release Op$Table Resource */
                CALL RQ$RECEIVE$CONTROL (Common$Info$Region, @Exception);
                CALL Get$New$Appl;
                IF (Status <> Successful)
                  THEN  CALL RQ$SEND$CONTROL (@Exception);  /* Release Common$Info Resource */

                  ELSE  /* Continue as long as successful */
                    DO;
                      CALL RQ$RECEIVE$CONTROL (Op$Table$Region, @Exception);

                      /* Ensure that while the Op$Table resource was released, that another
                          task did not attach the application.  If this occurred, then the
                          procedure must be restarted since control is in the wrong option */

                      CALL Op$Table_Search$Appl (Rqst$Appl, @Op$Appl$Num, Status$Ptr);
                      IF (Status = Found)
                        THEN
                          DO;
                            CALL RQ$A$DELETE$CONNECTION (New$Appl.Appl$Connection, No$Mailbox, @Exception);
                            CALL RQ$SEND$CONTROL (@Exception);  /* Release Op$Table Resource */
                            CALL RQ$SEND$CONTROL (@Exception);  /* Release Common$Info Resource */
                            GOTO RESTART;
                          END; /* Restart from outermost level */

                      /* ELSE the above did not occur - Continue & Attach the Application to the Op$Table */

                      CALL Op$Table_Attach$Appl (Rqst$Appl, @Op$Appl$Num, Status$Ptr);
                      IF (Status <> Successful)
                        THEN  /* No more Application entries are available in the Op$Table */
                          CALL RQ$A$DELETE$CONNECTION (Reqd$Appl.Appl$Connection, No$Mailbox, @Exception);
                        ELSE
                          DO;
                            Reqd$Op$Appl$Ptr = @Op$Appl$Map(Op$Appl$Num);
                            IF (Reqd$Appl.All$Reqd OR Whole$Appl$Rqst)
                              THEN CALL Load$All$Disks;
                              ELSE CALL Load$Single$Disk;
                          END; /* Else */

                      /* If errors have occured then resources are still being accessed */
                      IF (Status <> Successful)
                        THEN
                          DO;
                            CALL RQ$SEND$CONTROL (@Exception);  /* Release Op$Table Resource */
                            CALL RQ$SEND$CONTROL (@Exception);  /* Release Common$Info Resource */
                          END;
                    END; /* Else Continue */

              END; /* Case where Application is not immediately available & has to be loaded */

          END; /* Reqd$Appl <> Non$Appl */

        END Common$Disk$Open;

    /* END COMMON$DISK$OPEN PROCESS */

    /* ------------------------------------------------------------------------------------ */
```

REFERENCES

1. De Souza, E.A. *EDULAN: A Local Area Network for an Educational Environment*, MSc Dissertation, University of the Witwatersrand, Johannesburg, S.A., 1987.

2. Schutz, R.A. *An Investigation into the Automatic Analysis of Pascal Programs*, MSc Project Report, University of the Witwatersrand, Johannesburg, S.A., 1987.

3. Svobodova, L. "File Servers for Network-based Distributed Systems", *ACM Computing Surveys*, Vol 16, No.4, Dec 1984, pp353-398.

4. Hurst, M. "Sharing data between Microcomputers on a local Net", *Data Communications*, Sept 1984, pp139-148.

5. Miller S. and Collins M. "Toward a Reference Model of Mass Storage Systems", *IEEE Computer*, July 1985, pp9-22.

6. Birrell A.D. and Needham R.M. "A Universal File Server", *IEEE Trans. on Software Engineering*, Vol SE-6, No.5, Sept 1980, pp450-453.

7. Hudson, C. "A Microprocessor File Server", *Software and Microsystems*, Vol 2, No.1, Feb 1983, pp9-13.

8. Rolander T., Baird R. and Wharton J. "Network Software borrows design from Microcomputer Operating System", *Data Communications*, Dec 1982, pp123-133.

9. Kolod, M. "IBM PC Disk Performance and the Interleave Factor", *Byte*, Fall 1985, pp283-290.

10. Shiell J. and Markoff J. "Adding a Hard Disk to your PC AT", *Byte*, Fall 1985, pp159-172.

11. Calingaert, P. *Operating System Elements: A user Perspective*, Prentice-Hall, 1982.

12. Walker, B.J. *Computer Security and Protection Measures*, Dawden, Hutchinson and Ross Inc., Pennsylvania, 1977.

13. Haskett, J.A. "Pass-Algorithms: A user validation scheme based on knowledge of secret algorithms", *Communications of the ACM*, Vol 27, No. 8, Aug 1984.

14. Jones, R. "Disc Cache for PCs", *Data Processing*, Vol 28, No.2, March 1986, pp79-81.

15. McKeon, B. "An Algorithm for Disk Caching with Limited Memory", *Byte*, Sept 1985, pp129-138.

16. Walker, A.J. *Structured Information Processing System Design (Rev 4.0)*, Dept. of Elec. Eng., University of the Witwatersrand, Johannesburg, S.A., 1986.

17. Booch, G. "Object-Oriented Development", *IEEE Transactions on Software Engineering*, Vol SE-12, No. 2, Feb 1986, pp211-221.

18. Bishop, J.M. *Data Abstraction in Programming Languages*, Computer Science Dept., University of the Witwatersrand, Johannesburg, S.A., 1984.

19. Tucker, S. "iRMX86 - A Realtime operating system for advanced 16-bit Microcomputers", *Microprocessors and Microsystems*, Vol 7, No. 8, Oct 1983.

20. *Introduction to the iRMX 86 Operating System*, Intel Corporation, Santa Clara, California, 1982.

21. *iRMX 86 Nucleus Reference Manual*, Intel Corporation, Santa Clara, California, 1982.

22. *PL/M-86 User's Guide*, Intel Corporation, Santa Clara, California, 1982.

23. Voydock V. and Kent S., "Security Mechanisms in High-Level Network Protocols", *ACM Computing Surveys*, Vol 15, No. 2, 1983, pp135-171.

24. Voydock V. and Kent S., "Security in High-Level Network Protocols", *IEEE Communications Magazine*, Vol 23, No. 7, July 1985, pp12-24.

25. Fidlow D. and Waterhouse P. "A Comprehensive approach to Network Security", *Data Communications*, April 1985, pp195-213.

26. *LAN Software Report*, Novell Inc., Provo, Utah, USA, 1985.

27. *LAN Operating System Report*, Novell Inc, Provo, Utah, USA, 1986.

Author Hildyard Mark William
Name of thesis Design Of A Disk-based File Server Involving The Mapping Of "pseudo-disks". 1989

PUBLISHER:

University of the Witwatersrand, Johannesburg
©2013

LEGAL NOTICES:

Copyright Notice: All materials on the University of the Witwatersrand, Johannesburg Library website are protected by South African copyright law and may not be distributed, transmitted, displayed, or otherwise published in any format, without the prior written permission of the copyright owner.

Disclaimer and Terms of Use: Provided that you maintain all copyright and other notices contained therein, you may download material (one machine readable copy and one print copy per page) for your personal and/or educational non-commercial use only.

The University of the Witwatersrand, Johannesburg, is not responsible for any errors or omissions and excludes any and all liability for any errors in or omissions from the information on the Library website.