INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION

INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
34100 TRIESTE (ITALY) - P.O.B. 586 - MIRAMARE - STRADA COSTIERA 11 - TELEPHONE: 2240-1
CABLE: CENTRATOM - TELEX 460392-1

SECOND SCHOOL ON ADVANCED TECHNIQUES
IN COMPUTATIONAL PHYSICS
(18 January - 12 February 1988)

SMR.282/ 7

VMS

M.MATIELLO
Quantum Chemistry Group, University of Uppsala, Sweden

VAX/VMS overview

* Virtual Address eXtension
* Virtual Memory System
* 250 units in the Swedish University NETwork (300 next spring).
* Same operating system, same instruction set for all VAX/11 models.
* A multiuser environment.
* Process: An environment in which to edit & run.
* DCL (Command Language Interpreter): The image run by a process after login.
* DCL invokes execution of other programs.
* System images (editor, linker, file manager, communication, help,...).
* User images (your command-files, your programs).

* Users work with files: `edit`, `copy`, `type`, `open`, `read/write`, `print`, `compile`, `run`
* File access: By name.

```
dev:[dir.subdir]filename.extension;version
```

* Device: The recording media where the file is kept (disk, tape).
* Directory: A bookkeeping facility to group related files.
* Filename: The name.
* Extension: Some extensions are assummed by default when invoking system images.
* Version: Bookkeeping for file updating.

File management

CREATE *file*	cat > *name*
CREATE/DIR	mkdir
COPY(APPEND)	cp
DELETE	rm
DIFFERENCES	diff
DIRECTORY	ls
RENAME	mv
SEARCH	grep
TYPE	cat

OPEN, CLOSE, READ, WRITE

Editors (EDIT or EDIT/TPU).

(Write a DCL command in UNIX)

Editors

* Terminal dependent (VT compatible).
* Keypad (EDIT) / The "DO" key (TPU).

  MOVE (*up, down, right, left, by word, by line*)

  NEXT SCREEN, PREVIOUS SCREEN

  TOP, BOTTOM

  FIND (*string*)

  REPLACE (*'oldstring' 'newstring'*)

  DELETE (*character/word/line*)

  RESTORE (*word/line*)

  SELECT, REMOVE, INSERT (*a segment*)

  INCLUDE (*another file*)

  INSERT/OVERSTRIKE
  FORWARD/REVERSE
  SAVE, EXIT, QUIT

* TPU allows to preset PF- and <CTRL>- keys to perform editor functions.
* TPU allows to write a personal editor.

System information/performance

- HELP: Description of all VMS commands with all parameters and qualifiers.
- MONITOR: Updated system performance.
- SHOW (SET): *default, devices, logical, memory, process, protection, queue, symbol, system, time, users,...*

  SET CONTROL, SET HOST, SET PASSWORD
- FINGER
- Monitoring a program:
  SHOW PROCESS (/ALL, /CONTINUOUS)
  SHOW SYSTEM, SHOW STATUS

Communication

- MAIL — Extension: ".MAI"
- PHONE: Terminal-to-terminal.
- "SEND" (MSG)
- DECNET

Run applications

* "@" — Extension: ".COM"
* FORTRAN/LINK/RUN
  Extensions: ".FOR", ".OBJ", ".EXE"
* Other languages have similar commands and extensions.
* Object files created with different languages can be linked together.
* SPAWN subprocesses
* Batch operation:
  Batch queues.
  SUBMIT, PRINT

Logical names & symbols

* Associate a name with a device, a file, or another logical name.
* Logical tables.
* Give local names to devices.
* Give standard names to system files.
* Give VMS names to input/output files of user application programs (fortran files 5 & 6).
* Allows to operate independently of the actual device.
* `ASSIGN, DEFINE, DEASSIGN`
* Symbols: DCL alias.


```
(LNM$PROCESS_TABLE)
  "SYS$COMMAND" = "_KVAX3$LTA39:"
  "SYS$DISK" [super] = "WASA:"
  "SYS$DISK" [exec] = "WASA:"
  "SYS$ERROR" = "_KVAX3$LTA39:"
  "SYS$INPUT" = "_KVAX3$LTA39:"
  "SYS$OUTPUT" [super] = "_KVAX3$LTA39:"
  "SYS$OUTPUT" [exec] = "_KVAX3$LTA39:"
  "TT" = "_LTA39:"

(LNM$JOB_80349480)
  "SYS$LOGIN" = "WASA:[MARIO]"
  "SYS$LOGIN_DEVICE" = "WASA:"
  "SYS$SCRATCH" = "WASA:[MARIO]"

(LNM$GROUP_000024)
  "DSK" = "WASA:"
  "HLP$LIBRARY" = "DSK:<TEX.HLP>LASER"
  "HLP$LIBRARY_1" = "DSK:<SYSMARIO.HLP>MOPAC"
  "LIST" = "DSK:[JORGEN.TEOGRP.UTILITIES]LIST.EXE"
  "QLSR" = "DSK:<ALEJANDRO.PLOT.EXE>QLSR.EXE"
  "QQ" = "DSK:<ALEJANDRO.PLOT.EXE>QQ.EXE"
  "QSPLINE" = "DSK:<ALEJANDRO.PLOT.EXE>QSPLINE.EXE"

(LNM$SYSTEM_TABLE)
  "ACP$BADBLOCK_MBX" = "MBA4:"
  "DBG$INPUT" = "SYS$INPUT:"
  "DBG$OUTPUT" = "SYS$OUTPUT:"
  "DISK$KVAX3_SYS" = "KVAX3$DUA0:"
  "FINGERSHR" = "SYS$SYSTEM:FINGERSHR"
  "KVPLOT" [super] = "LTA2:"
  "KVPLOT" [exec] = "KVPLOT:"
  "LASERPRINTER" [super] = "LTA1:"
  "LASERPRINTER" [exec] = "LASERPRINTER:"
  "LNK$LIBRARY_2" = "WASA:[LIB]NAG.OLB"
  "LOKE" = "KVAX3$DUB0:"
  "MNT" = "LOKE:[SCRATCH.]"
  "MOM$LOAD" = "KVAX3$DUA0:[SYS0.SYSCOMMON.][MOM$SYSTEM]"
  "MTHRTL" = "SYS$SHARE:UVMTHRTL.EXE"
  "SCR" = "KVAX3$DUC1:"
  "SYS$ANNOUNCE" = "Welcome to kvaCluster, running VMS V4.6     on node KVAX3"
  "SYS$COMMON" = "KVAX3$DUA0:[SYS0.SYSCOMMON.]"
  "SYS$DISK" = "KVAX3$DUA0:"
  "SYS$ERRORLOG" = "SYS$SYSROOT:[SYSERR]"
  "SYS$EXAMPLES" = "SYS$SYSROOT:[SYSHLP.EXAMPLES]"
  "SYS$HELP" = "SYS$SYSROOT:[SYSHLP]"
  "SYS$INSTRUCTION" = "SYS$SYSROOT:[SYSCBI]"
  "SYS$LIBRARY" = "SYS$SYSROOT:[SYSLIB]"
  "SYS$MAINTENANCE" = "SYS$SYSROOT:[SYSMAINT]"
  "SYS$MANAGER" = "SYS$SYSROOT:[SYSMGR]"
  "SYS$MESSAGE" = "SYS$SYSROOT:[SYSMSG]"
  "SYS$NODE" = "KVAX3::"
  "SYS$SHARE" = "SYS$SYSROOT:[SYSLIB]"
  "SYS$SPECIFIC" = "KVAX3$DUA0:[SYS0.]"
  "SYS$SYLOGIN" = "SYS$MANAGER:SYLOGIN.COM"
  "SYS$SYSDEVICE" = "KVAX3$DUA0:"
  "SYS$SYSDISK" = "SYS$SYSROOT:"
  "SYS$SYSROOT" = "KVAX3$DUA0:[SYS0.]"
        = "SYS$COMMON:"
  "SYS$SYSTEM" = "SYS$SYSROOT:[SYSEXE]"
  "SYS$TEST" = "SYS$SYSROOT:[SYSTEST]"
  "SYS$TOPSYS" = "SYS0"
  "SYS$UPDATE" = "SYS$SYSROOT:[SYSUPD]"
  "SYS$WELCOME" = "@SYS$MANAGER:WELCOME.TXT"
  "WASA" = "KVAX3$DUB1:"
```

VMS internals

* System design.
* System architecture.
* Process characteristics.
* Disk structure.
* The linker.
* Run-time libraries.

System design

⋆ SYSGEN:

Ca. 220 system parameters.
Adapt the System to local needs.
Maximum number of processes.
Maximum primary memory to be used.
Maximum working set.

⋆ STARTUP.COM

INSTALL system programs
DEFINE symbols/logical (system/group)
MOUNT devices
Start network (DECNET)
Start local applications (ex.: FINGER)
Start queue manager.

⋆ SYSUAF.DAT Database with all usernames, default directories, user parameters, etc.

⋆ NET*.DAT Databases with network info: Remote nodes, network objects, proxy list, etc.

* Protection/access mode: Determines user rights to `read`, `execute`, `write` and `delete` files.
  Protects system and user from user errors.

* Privileges: Ability to alter the environment.
  Protects the system from user errors.

* Priority: A scheduling mechanism.

- `INSTALL`

  Decreases image activation time.

  Allows to run privileged code (`/PRIV`).

  `/SHARE`: Only one copy in primary memory.

  `/OPEN`: Eliminates directory search (permanent channel to the file).

  `/HEADER`: File header resident in primary memory (saves one `I/O` per file access).

System architecture

- Single processor.

- Many processors.

  Tightly coupled.

  Loosely coupled

  Clustered

  Networked.

- Tightly coupled:
  Processors do not operate independently.
  Run the same copy of the operating system.
  Hierarchy. Shared memory.

- Loosely coupled:
  Operate independently.
  Run own copy of operating system.
  Local and shared memory.
  User synchronization.

- Cluster:
  Processors are independent.
  Run own copy of operating system.
  Shared cluster-wide devices (speed).
  One system disk.
  User can run applications on the other processors (via batch).
  Experienced as a single system.

- Networked: `DECNET`.

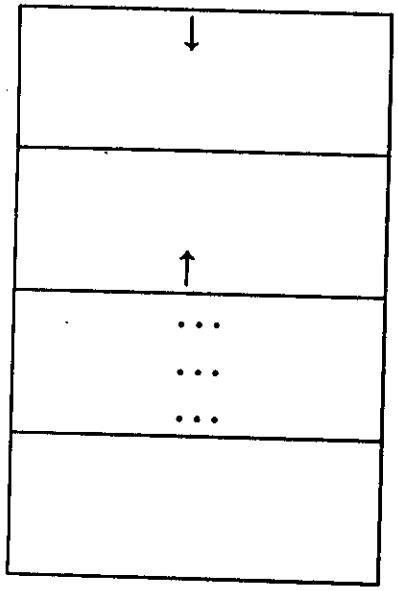
Structure of a process

* Login → DCL
* Execute images
* Logout

⋆ Hardware context: The status of the hardware at a given moment (Saved by the system when the process is not executing).

⋆ Software context: Information used by the system to control process activities.

⋆ Virtual address space: The range of memory addresses known to a process. Used by a program to identify the location of instructions and data.



Process
Program region

Process
Control region

System region

Reserved region

* The Process region is specific to each process.
* The System region is shared by all processes.

Executing an image

- Programs are stored on disk.
- At run-time, instructions and data are copied to primary memory.
- Information is copied in 512B *pages*.
- Resident pages constitute the *working set*.
- Paging: When the application is larger than the *working set*. (`PAGEFILE.SYS`).
- Try to arrange your applications contiguously.
- `SYSUAF.DAT` controls maximum working set and paging.
- `SWAPPING`: Process working set is dumped into an auxiliary file `SWAPFILE.SYS`.

- Too much swapping: Increase primary memory?
- Too much paging: Re-write your application?

Disk structure

* Initialize disk.

* `INDEXF.SYS`: Boot block, Home block, Backup of home block, Backup of file header, index file bitmap, file headers (index file, all other files).

* `BITMAP.SYS`: All available areas.

* `BADBLK.SYS`: Regions unsuitable for use.

⋆ Boot block: In the system disk, loads the operating system.

⋆ Home block

  Disk volume information

  Label, owner, protection

  Initial entry point into file structure.

⋆ Bit map: 1 bit per file header. Controls the number of files in the volume.

⋆ File header:

  Access validity & identification.

  Map area: Pointers to blocks allocated to the file.

  `DUMP/HEADER` *filename.*

⋆ Block allocation:

Minimum allocation: *cluster*

Contiguous clusters are called *extents.*

Header pointers use *extents.*

Large clusters: Wasted space.

Small clusters: Too much fragmentation

⋆ Deletion:

Eliminate file header

Mark the extents as available in `BITMAP.SYS`.

Directory structure

∘ Associate symbolic file name with file identity.

∘ Pointer to the file header area in `INDEXF.SYS`

∘ `SET DEFAULT`: Work within the context of a directory.

∘ Information is stored in a `".DIR"` file.

Linker

* Input: Object files, library files.
* Output: Executable images, shareable images.
* Sequence:

  Collect input files.

  Organize input into groups (`clusters`).

  Assign virtual address to image sections.

  Write clusters to image file.

⋆ To reduce paging: Link with the `/CLUSTER` option.

Contiguous pages should have contiguous code.

Run-time libraries
System services

- Shareable code accessible at run-time.
- Available to all images regardless of the language used to compile the program.
- Run-Time libraries:

  DTK$: DECTalk.

  LIB$: General purpose:
  `LIB$CREATE_DIR LIB$DAY_OF_WEEK`
  `LIB$DELETE_SYMBOL LIB$DELETE_LOGICAL`
  `LIB$GET_INPUT LIB$SPAWN LIB$WAIT`
  `LIB$FIND_FILE LIB$FIND_FILE_END`
  `LIB$DO_COMMAND LIB$STOP`

  MTH$: Arithmetic in H-precision.

  OTS$: Number ↔ String conversions, Complex arithmetic, etc.

SMG$: Screen manager.

STR$: String operations.

UIS$: Graphics.

- System services: SYS$:

  `$ASCTIM $BINTIM $CRELNT $HIBER $QIO $QIOW $SCHDWK $SETAST $SETPRV $WAKE`

* Usage: `INCLUDE '($SSDEF)'` or `'($SMGDEF)'`.

* Set parameters to desired values (manual).

* Use as `SUBROUTINE` or as `FUNCTION`.

* Return status (even values indicate errors).

* Arguments: `X=1, T='1'`

  By Value. Pass the number 1: `%VAL(X)`.

  By Reference. Pass an address: `X`
  (Standard in `FORTRAN` for numbers).

  By Descriptor. Pass address of a descriptor: `T`
  (Standard in `FORTRAN` for characters).

* An example of linking code written with different languages.


```
      program hiberwake
c
c  This program performs the following tasks;
c  1.  Find the name of the terminal where it was started.
c    (This data is necessary for step 2.)
c  2.  Create a subprocess, in an hibernation state.
c  3.  Wake up the subprocess.
c  4.  Schedule a wake up for the process.
c  5.  Hibernate.
c  6.  Kill the subprocess after waking up.
c
      implicit integer*4 (a-z)
      include '($prcdef)'
      include '($ssdef)'
      include '($lnmdef)'
      character name*4/'subp'/,image*6/'subpro'/,
     1  later*6/'0 ::15'/
      character*255 terminal/'SYS$OUTPUT'/,ret_string
      character*17 tabnam/'LNM$PROCESS_TABLE'/
      integer*2  bin_later(2),bin_now(2),trnlst(20)/20*0/
      integer*4  ret_length/10/,ret_attrib
      byte esc_null_num(2)/'1B'X, '00'X/
      character*2 esc_null
      equivalence (esc_null,esc_null_num)
c
c  set up trnlst
c
      trnlst(1)=255
      trnlst(2)=lnm$_string
      trnlst(3)=%LOC(ret_string)
      trnlst(7)=4
      trnlst(8)=lnm$_attributes
```

```
      trnlst(9)=%LOC(ret_attrib)
      trnlst(13)=4
      trnlst(14)=lnm$_length
      trnlst(15)=%LOC(ret_length)
c
c  Find terminal name.  Trnlnm translates the logical name
c  sys$output.  If this translation is also a logical
c  name, repeat the process.
c
100   status=sys$trnlnm(,tabnam,terminal(1:ret_length)
     1   ,,trnlst)
      if(.not.status) call lib$stop(%val(status))
      if(iand(lnm$m_terminal,ret_attrib).eq.0) then
        terminal=ret_string(1:ret_length)
        go to 100
      endif
c
c  Now ret_string has the name of the terminal, probably
c  preceeded by some escape characters.
c
      if(ret_string(1:2).eq.esc_null)then
        ret_string=ret_string(5:ret_length)
        ret_length=ret_length-4
      endif
c
c  Create the subprocess.
c
      write(*,'(a,a)') ' Creating subprocess ',name
      write(*,'(a,a)') ' from ',ret_string(1:ret_length)
      call lib$wait(2.0)
      state=sys$creprc(pid,image,,ret_string(1:ret_length),
     1  ret_string(1:ret_length),,,name,%val(3),,,
```



```
     2  %val(prc$m_hiber))
      if(.not.state)call lib$stop(%val(state))
      write(*,'(a,a,a)') ' Subprocess ',name,' created.'
c
c  Awake the subprocess.
c
      write(*,'(a)') ' Waking-up subprocess.'
      call lib$wait(2.0)
      state=sys$wake(pid,name)
      if(.not.state) call lib$stop(%val(state))
      write(*,'(a)') ' Subprocess is awake.'
c
c  Schedule an own wake-up
c
      write(*,'(a)') ' Schedule an own wake-up.'
      call sys$bintim(later,bin_later)
      state=sys$schdwk(,,bin_later,)
      if(.not.state) call lib$stop(%val(state))
      write(*,'(a)') ' Wake-up successfully scheduled.'
c
c  Hibernate
c
      write(*,'(a)') ' Hibernation starts.  See you later.'
      call sys$hiber()
c
c  Kill subprocess after waking up.
c
      write(*,'(a)') ' Main process awake.'
      call lib$wait(2.0)
      call sys$delprc(pid,name)
      write(*,'(a)') ' ** The subprocess was deleted.  **'
      end
```

```
      program subpro
c
c     This is the image run by the subprocess subp according
c     to program hiberwake.
c
      implicit integer*4 (a-z)
      i=0
      do while(1.gt.0)
      i=i+1
      write(*,'(a,i3,a)')' This is subprocess cycle number ',i
      call lib$wait(0.5)
      enddo
      end
```



```
      program synchron
c
c     This program performs the following tasks:
c     1.  Allocate an event flag (a communication register).
c     2.  Call a routine that starts a subprocess.
c     3.  Show time.
c     4.  Loop until the flag is reset by the subprocess.
c     5.  Show time and end.
c
      implicit integer*4 (a-z)
      include   '($ssdef)'
      integer*4   elap(2)
      character*23   ascii_time
      character*6  elapsed/'0 ::10'/
c
c     Allocate and clear event flag.
c
      status=lib$get_ef(flag1)
      if(.not.status) call lib$stop(%val(status))
      status=sys$clref(flag1)
      if(.not.status) call lib$stop(%val(status))
c
c     Call subroutine
c
      call ast
c
c Print the current time.
c
      status=sys$asctim(,ascii_time,,)
      if(.not.status) call lib$stop(%val(status))
c
c     Convert ascii to binary time and set a timer.
```

```
c
  status=sys$bintim(elapsed,elap)
  if(.not.status) call lib$stop(%val(status))
  status=sys$setimr(%val(flag1),elap,,%val(2))
  if(.not.status) call lib$stop(%val(status))
c
c  Read status of flag and loop until flag is set.
c
  status=sys$readef(%val(flag1),icond)
  do while (status.ne.ss$_wasset)
  status=sys$readef(%val(flag1),icond)
  call sys$setast(%val(0))
  write(*,'(a)')' Main program is waiting for the flag.'
  call sys$setast(%val(1))
  call lib$wait(0.5)
  enddo
c
c  Print time and exit.
c
  status=sys$asctim(,ascii_time,,)
  if(.not.status) call lib$stop(%val(status))
  stop'bye'
  end
  subroutine ast
  implicit integer*4 (a-z)
  integer*4   bi(2)
  character*5  interval/'0 ::3'/
  external truesub
c
c  Convert ascii to binary time.
c
  status=sys$bintim(interval,bi)
```



```
  if(.not.status) call lib$stop(%val(status))
c
c  Set the true routine to work.
c
  status=sys$setimr(,bi,truesub,%val(1))
  if(.not.status) call lib$stop(%val(status))
  return
  end

  subroutine truesub
  implicit integer*4 (a-z)
  call ast
  do i=1,3
  write(*,'(a)')' The ast is writing.'
  enddo
  return
  end
```