OpenVMS Programming Concepts and Interfaces (2D05)

IT Symposium 2004

Thilo Lauer
Technical Consultant OpenVMS

Agenda

- What‘s in a name?
- Overview of OpenVMS APIs
- Classes of System Services
- Structure of System Services
- Examples (Mailboxes, Terminal I/O)

*** WARNING ***

This presentation is intentionally…

- imprecise
- incomplete
- scratching the surface

… and in the end, you'll definitely have more questions than answers!

Overview of OpenVMS APIs

- General Purpose Library functions
  - LIB
- Utility Routines
  - MTH Mathematics
  - SMG Screen Management
  - MAIL MAIL Interface
  - FDL File Description Language API
  - OTS, STR Conversion, mathematical Routines
  - PPL Parallel Processing Library
  - EDT/TPU callable Editors
  - DCX Data Compression
  - CLI Command Language Interface
  - LBR, NCS, PSM, SOR, SMB, LGI, CONV, UTIL, BACKUP...

Overview of OpenVMS APIs

- Special APIs
  - DTK DECTalk Interface
  - CPML Compaq Portable Mathematics Library
  - LDAP Lightweight Directory Access Protocol
  - PTHREADS POSIX Threading package
  - SSL Secure Socket Layer
  - SSH Secure Shell
- System Services
  - SYS access the heart of OpenVMS

Classes of System Services

- creating/controlling/deleting processes
  - `$CREPRC` create Process
  - `$DELPRC` delete Process
  - `$FORCEX` forced image/process exit
  - `$SUSPND` suspend process
  - `$RESUME` resume process
  - `$HIBER` hibernate process
  - `$GETJPI` get job/process information
  - `$SETPRV` set process privileges
  - `$SETPRI` set process priority
  - `...`

Classes of System Services

- memory management
  - `$CRETVA` create virtual address space
  - `$DELTVA` delete virtual address space
  - `$CRMPSC` create and map section
  - `$MGBLSC` map to global section
  - `$EXPREG` expand memory region
  - `$LKWSET` lock page in working set
  - `$PURGWS` purge working set
  - `$SETSWM` set process swapping mode
  - `$CREATE_BUFOBJ` create buffer object
  - ...

Classes of System Services

- synchronizing events
  - `$SYNCH` synchronize on EFN+IOSB status
  - `$WAITFR` wait for specific Event Flag
  - `$WFLOR` wait for logical OR of EFNs
  - `$WFLAND` wait for logical AND of EFNs
  - `$CLREF` clear Event Flag
  - `$SETEF` set Event Flag
  - ...

Classes of System Services

- resource sharing
  - `$ENQ` queue lock/conversion request
  - `$DEQ` release/cancel lock request
  - `$GETLKI` get lock information
  - `$START_TRANS` start transaction
  - `$END_TRANS` end transaction

➔ see also last year's talk about the OpenVMS Distributed Lock Manager (1D05)!

Classes of System Services

- general I/O
  - `$QIO(W)` queue I/O request
- RMS I/O
  - `$OPEN` open a file
  - `$GET, $PUT` read/write a record
  - `$READ, $WRITE` read/write blocks
  - `$SEARCH` search for a file
  - `$CONNECT` connect a stream
- File System related I/O (ACP QIO)
  - `$QIO(W)` with special function codes

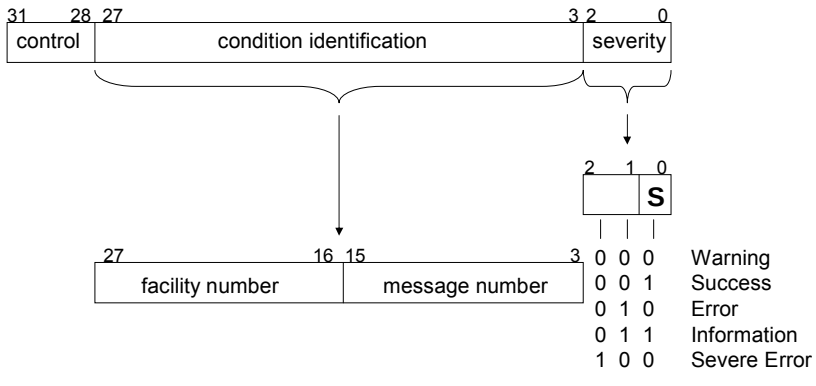
Classes of System Services

- condition handling
  - `LIB$SIGNAL` signal a software condition
  - `LIB$STOP` same, but stop execution
  - `LIB$ESTABLISH` establish condition handler
  - `$UNWIND` unwind Call chain on stack
  - `$DCLEXH` declare Exit Handler
  - ...

Names…

- Condition Value…

```
status = SYS$Service(par1, par2...)
```

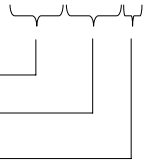


Names…

- …Condition Value, Example

```
Condition Value =          00D38094  (hex)

Facility   : LOGIN
Message    : BADDAY
Severity   : Severe
```



Resulting message:

```
%LOGIN-F-BADDAY, you are not authorized to login today
```

Names…

- …Condition Value, checking for success

call the service

```
status = SYS$Service(par1, par2...)
```

possible ways to check the condition value

- `if (status != SS$_NORMAL) exit`
- `if (!(status & 1)) lib$signal(status)`
- `if (not (odd(status))) sys$exit(status)`
- `IF (.NOT. STATUS) CALL LIB$SIGNAL(%VAL(STATUS))`

…Names

- Parameter passing
  - ***by value***

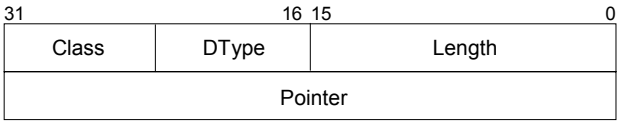
```
status = SystemService(value(42))
```

  - ***by reference***

```
status = SystemService(addr(42))
```

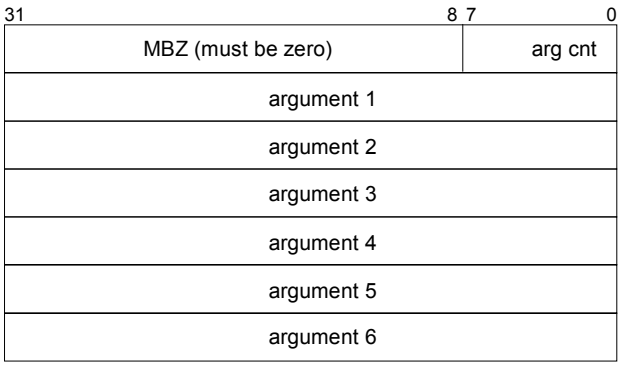
  - ***by descriptor***

```
status = SystemService(addr(descriptor(42)))
```



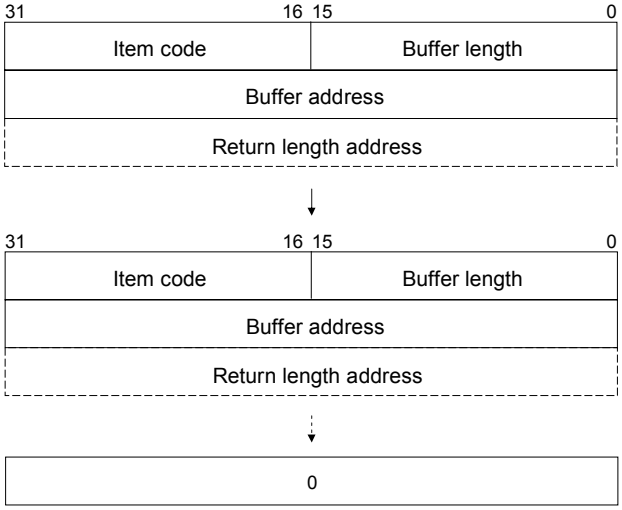
…Names

- Parameter list



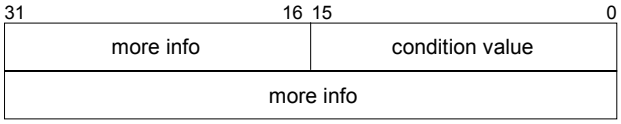
…Names

- Item List



…Names

- I/O Status Block (IOSB)



...Names

- Symbolic Constant Naming Conventions

  General format is ***facility$type_name***

  - ***facility*** describes class, module, function
  - ***$*** fixed part of symbolic name
  - ***type***
    - `V` Single Bit Value
    - `K, C` Constant
    - `T` Text Field
    - `M` Bit Mask
    - `S` Size Field
    - `B,W,L,Q` Length of Field
    - `nothing` Item Code, Field Name, Status Code

...Names

- Symbolic Constant Naming Conventions
  - Examples

  We call `SYS$GETSYI` with Item Code `SYI$_HW_MODEL`. Getting `VAX$K_V730` means we're using a VAX 11/730.

  Opening a file with `SYS$OPEN`, specifying `FAB$B_FAC=FAB$M_PUT`, requests write access to the file.

  - All these symbolic names are defined in `[$]facilityDEF,` which you include/inherit as necessary.

...Names

- Symbolic Constant Naming Conventions

Mixing up V_ and M_ symbolic constants is the single most common source of errors when programming with OpenVMS!

Example:
The number of the bit requesting the Deferred Write option for a file is `FAB$V_DFW (5 hex, 0...0101 bin)`

But: `FAB$B_FOP = FAB$V_DFW` results in `0...0101`, which means Bits 0+2 (`FAB$M_ASY`) are set!

Result: no Compiler or Linker error: your program simply fails!

Solution: `FAB$B_FOP = FAB$M_DFW`

Structure of System Services

- synchronous Form
  `SYS$QIOW(), SYS$ENQW(), SYS$GETSYIW()…`
- asynchronous Form
  `SYS$QIO() , SYS$ENQ() , SYS$GETSYI()…`

`SYS$synchron() = SYS$SYNCH() + SYS$asynchron()`

`SYS$SYNCH(EFN, IOSB) =`

- `SYS$WAITFR(EFN)`
- Test if IOSB <> 0
  - yes return
  - no clear EFN and wait again

Structure of System Services

- Event Flags

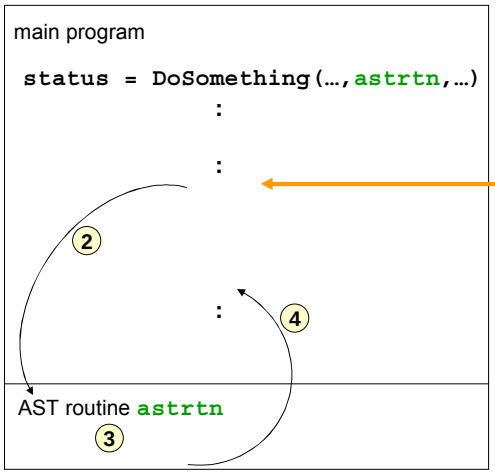
```
status = DoSomething(…,EFN=5,…)
              :
              :
              :
status = wait(EFN=5)
```

OpenVMS

1. System Service gets executed
2. Event Flag will be set

Structure of System Services

- Asynchronous System Traps (ASTs)



OpenVMS

1. System Service gets executed
2. AST Routine gets called

Example 1 - ***Repeating Alarm Clock***

Requirements:

- Generate Alarm at given absolute time
- Repeat Alarm every minute (5 times)

Implementation:

- via process hibernation and scheduled wakeup request

Example 1 - ***Repeating Alarm Clock***

Code snippets:

- *define two descriptors to hold our time strings*

```
char TimeStr[] = "14-APR-2004 12:20:00.00";
$DESCRIPTOR(Alarm, TimeStr);
$DESCRIPTOR(Interval, "0 :01:00");
```

Example 1 - ***Repeating Alarm Clock***

Code snippets:

- *convert time strings to OpenVMS time format*

```
if (((Status=SYS$BINTIM(&Alarm,DayTime)) &1) != 1)
   LIB$STOP(Status);

if (((Status=SYS$BINTIM(&Interval,RepTime)) &1) != 1)
   LIB$STOP(Status);
```

- *Tell OpenVMS how to schedule our process*

```
if (((Status=SYS$SCHDWK(0,0,&DayTime,&RepTime)) &1) !=1)
   LIB$STOP(Status)
```

Example 1 - ***Repeating Alarm Clock***

Code snippets:

- *manage the repeating alarm function*

```
SYS$HIBER();

LIB$INIT_TIMER();
LIB$DATE_TIME(&Alarm);
printf("Alarm triggered at: %s\n", TimeStr);

for (i=1; i<=5; i++) {
   SYS$HIBER();
   LIB$SHOW_TIMER(0, &Code);
}
```

Example 1 - ***Repeating Alarm Clock***

Result:

```
Alarm triggered at: 14-APR-2004 12:20:00.00
 ELAPSED TIME =     0 00:01:00.00
 ELAPSED TIME =     0 00:02:00.00
 ELAPSED TIME =     0 00:03:00.00
 ELAPSED TIME =     0 00:04:00.00
 ELAPSED TIME =     0 00:05:00.00
```

Example 2 – ***Advanced Terminal I/O***

Requirements:

- Get user input from terminal
- issue a prompt string

Implementation:

1. $QIOW
2. LIB$GET_INPUT

Example 2 – *Advanced Terminal I/O*

Code snippets:

- *define two descriptors to hold our time strings*

```
struct  iosb_struct {
  short       status;
  short       BytesRead;
  int         Dummy_Int;
} IOSB;

int  status, TermChannel;
char Buffer[100];
char Prompt[]="You have 5 seconds to enter something:";
$DESCRIPTOR(TermDesc, "TT");
```

Example 2 – *Advanced Terminal I/O*

- *Assign channel and issue $QIOW*

```
status = SYS$ASSIGN(&TermDesc, &TermChannel, 0, 0, 0);
if (!(status & 1)) lib$stop(status);

status = SYS$QIOW(3,           // Event flag
         TermChannel,          // Channel number
         IO$_READPROMPT        // I/O function code
           | IO$M_TIMED,       // I/O function modifier
         &IOSB,                // I/O status block
         0, 0,
         Buffer,               // P1: buffer
         sizeof(Buffer),       // P2: buffer length
         5,                    // P3: Timeout value
         0,
         Prompt,               // P5: Prompt Buffer address
         sizeof(Prompt));      // P6: prompt Buffer size
```

Example 2 – *Advanced Terminal I/O*

- *Perform error checking*

```
if (status & 1)    status = IOSB.status;

if (!(status & 1)) {
  if (status == SS$_TIMEOUT) {
    printf("\rToo late!\n");
    exit(1);
  }
  else
    lib$stop(status);
}
printf("You entered <%.*s>\n", IOSB.BytesRead, Buffer);
```

Example 2 – *Advanced Terminal I/O*

- *Results*

```
$> run qio
You have 5 seconds to enter something: OpenVMS is GREAT!
You entered <OpenVMS is GREAT!>

$> run qio
You have 5 seconds to enter something:
Too late!
$>
```

Example 2 – *Advanced Terminal I/O*

- *Alternate method via LIB$GET_INPUT*

```
void main (void) {
  int   status;
  short BytesRead;
  char  Buffer[100];
  $DESCRIPTOR(PromptDesc, "Enter something: ");
  $DESCRIPTOR(BufferDesc, Buffer);

  status = LIB$GET_INPUT(&BufferDesc,
                         &PromptDesc,
                         &BytesRead);
  if (!(status & 1)) lib$stop(status);

  printf("You entered <%.*s>\n", BytesRead, Buffer);
}
```

Example 2 – *Advanced Terminal I/O*

- *Results*

```
$> run get
Enter something: OpenVMS is still GREAT!
You entered <OpenVMS is still GREAT!>
$>
```

Resources

- Guide to Creating OpenVMS Modular Procedures
- OpenVMS Calling Standard
- Programming Concepts Manual
- RTL Library Manual
- Guide to File Applications
- System Services Reference Manual
- I/O Users Reference Manual
- + + + ...

Questions?