

Lists.hyper

COLLABORATORS

	<i>TITLE :</i> Lists.hyper		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		August 26, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Lists.hyper	1
1.1	List Reference (Wed Jul 15 14:43:30 1992)	1
1.2	List Reference : Introduction	3
1.3	List Reference : Summary of all available lists	3
1.4	List Reference : Using the 'list' command	4
1.5	List Reference : Using the 'info' command	4
1.6	List Reference : Using the 'for' command	5
1.7	List Reference : attc	6
1.8	List Reference : conf	7
1.9	List Reference : crsh	7
1.10	List Reference : dbug	8
1.11	List Reference : devs	9
1.12	List Reference : dosd	9
1.13	List Reference : exec	10
1.14	List Reference : fdfi	10
1.15	List Reference : fils	11
1.16	List Reference : font	12
1.17	List Reference : func	13
1.18	List Reference : graf	14
1.19	List Reference : ihan	14
1.20	List Reference : intb	15
1.21	List Reference : intr	15
1.22	List Reference : libs	16
1.23	List Reference : lock	16
1.24	List Reference : lwin	17
1.25	List Reference : memr	18
1.26	List Reference : moni	19
1.27	List Reference : port	19
1.28	List Reference : pubs	20
1.29	List Reference : pwin	20

1.30 List Reference : resm	21
1.31 List Reference : reso	22
1.32 List Reference : scrs	22
1.33 List Reference : sema	23
1.34 List Reference : stru	23
1.35 List Reference : task	24
1.36 List Reference : wins	25

Chapter 1

Lists.hyper

1.1 List Reference (Wed Jul 15 14:43:30 1992)

Contents:

Introduction

Summary of all available lists

Using the 'list' command

Using the 'info' command

Using the 'for' command

Lists:

attc

(attached keys or macros)

conf

(autoconfigs)

crsh

(crashed programs halted by PowerVisor)

debug

(all programs you are debugging)

devs

(exec devices)

dosd

(dos devices)

exec

(execbase structure)

fdfi

(all loaded fd-files)

files

(open files)

font
(open fonts)

func
(function monitor nodes)

graf
(graphicsbase structure)

ihan
(input handlers)

intb
(intuitionbase structure)

intr
(exec interrupts)

libs
(libraries)

lock
(locks)

lwin
(logical windows)

memr
(exec memory list)

moni
(monitors)

port
(message ports)

pubs
(public screens)

pwin
(physical windows)

resm
(resident modules)

reso
(resources)

scrs
(screens)

sema
(semaphores)

stru

(structure definitions)

task
(tasks and processes)

wins
(windows)

Various:

Back to main contents

1.2 List Reference : Introduction

PowerVisor can show you a lot of operating system lists. All the lists PowerVisor can show you (including the ones defined for PowerVisor) are explained in detail in this tutorial.

Warning! Each list has an entry in this chapter. The first line (after 'Name') is the name of the list as you would use it as an argument to the `list` or `info` commands.

So you need to respect the required uppercase part after 'Name'.

The second line (after 'Cmd') is the name of the command needed to go to that current list. Here you need to respect the uppercase part after 'Cmd'.

1.3 List Reference : Summary of all available lists

The following lists are available at this moment :

(All lists with a '*' have more information in the AmigaDOS 2.0 version, this (extra) information can be viewed with the `info` command or the `list` command)

Big structures :

Exec * the listing of the ExecBase structure
Intb IntuitionBase structure
Graf * Graphics base structure

Exec/dos/graphics and intuition things :

Task * The listing of the tasks in the system (default list)
Libs Exec-Libraries
Devs Exec-devices
Reso Exec-Resources
INTR Exec-Interrupts
Memr Memory list
Port Message ports
Wins * All windows
Scrs Screens
Font Fonts currently in memory
DOsd Dos-devices
SEma Semaphores
RESM Resident modules
Fils Open files
LOck Locks
IHan Input handlers

```

COnf          AutoConfigs
MOni *        Monitors (AmigaDOS 2.0 only)
PUBs *        Public Screens (AmigaDOS 2.0 only)
PowerVisor things :
FUnc          All Function monitor nodes (see addfunc command)
FDfi          All fdfiles loaded (see loadfd command)
Attc          All attached keys (see attach command)
Crsh          All crashed programs
DBUG          All debug nodes (see Debugging chapter)
STru          All structure defines (see addstruct command)
LWin          All logical windows for PowerVisor
PWin          All physical windows for PowerVisor

```

1.4 List Reference : Using the 'list' command

The `list` command can be used to show a list.

```
< list task <enter>
```

or

```
< task <enter>
```

```
< list <enter>
```

```

> Task node name      : Node      Pri StackPtr  StackS Stat Command      Acc
> -----
> Background Process : 07E28330 00 07E2D500   4096 Wait iprefs      (02) -
> RexxMaster          : 07E4DD38 04 07E4E57A   2048 Wait              (00) -
> PowerSnap 1.0 by Nic: 07E41B48 05 07E42392   2000 Wait              PROC -
> ...
> input.device        : 07E08B22 14 07E09B28   4096 Wait              TASK -
> trackdisk.device   : 07E0F1B4 05 07E0F3C6    512 Wait              TASK -
> RAM                 : 07E23BF8 0A 07E23EE6   1200 Wait              PROC -
> Background Process : 07E72728 04 07E8834E  12000 Run   pv          (05) -

```

(All fields in this list are explained in the following sections).

Using the `base()` function you can get the first element in the list :

```

< disp base() <enter>
> 07E28330 , 132285232

```

Using the `curlist()` function you can ask the current list. This function returns a pointer to the current list string. You can print this string with the `print` command :

```

< print \ (curlist(),%s)\0a <enter>
> task

```

1.5 List Reference : Using the 'info' command

The `info` command can be used to ask more information about an element of a list.

Let's assume we have the 'PowerVisor' task in the task list (this is in fact the case since you have probably started PowerVisor :-)

```
< info task:powervisor task <enter>
```

or

```
< info powervisor task <enter>
```

or

```
< info powervisor <enter>
```

or

```
< info task:powervisor <enter>
```

```
> Task node name      : Node      Pri StackPtr  StackS Stat Command      Acc
> -----
> PowerVisor1.0.task : 07E70370 00 07E7137E    4096 Wait           TASK -
>
> IDNestCnt          : 00          | TDNestCnt          : FF          | SigAlloc          : E000FFFF
> SigWait            : E0000000 | SigRecvd           : 00000000 | SigExcept         : 00000000
> TrapAlloc          : 8000          | TrapAble           : 0000          | ExceptData        : 00000000
> ExceptCode         : 00F83AEC | TrapData           : 00000000 | TrapCode          : 07E77696
> SpLower            : 07E703D0 | SpUpper            : 07E713D0 | SpReg             : 07E7137E
> MemEntry           : 07E703BA | UserData           : 00000000 |
```

The first version of the command is the safest one. There is nothing that can go wrong there.

The second version could crash if the current list is not equal to the 'task' list. This is because PowerVisor will then try to interpret the element starting with 'powervisor' in the other current list as a task. (If you are lucky there is no such element in the current list, in that case you simply get an error).

The third version is also safe although you could end up with the wrong information for the wrong element in the wrong list.

The last version could crash if the current list is not equal to the 'task' list. It is in fact the most dangerous version of all.

The `info` command does not work for the 'Exec', 'IntB' and 'Graf' lists. This is because these lists are structures and already give you all information there is.

1.6 List Reference : Using the 'for' command

If you want to execute a command for each element in a list you can use the `for` command (not for the 'Exec', 'Graf' and 'IntB' lists). The

command you supply as an argument for 'for' is executed with the pointer to the element in the 'rc' variable :

```
< for task disp rc <enter>
> 07E28330 , 132285232
> 07E4DD38 , 132439352
> 07E41B48 , 132389704
> 07E5B248 , 132493896
> ...
> 07E52140 , 132456768
> 07E08B22 , 132156194
> 07E23BF8 , 132267000
> 07E72728 , 132589352
```

Since the 'for' command remembers all output in memory and only starts printing after the list is traversed, you need not worry about the list becoming corrupt after a long time (This is especially true for the task list since this is a very busy list).

1.7 List Reference : attc

```
Name : 'Attc'
Cmd  : 'ATTC'

Some related commands :

    attach      : make a key attachement
    remattach   : remove a key attachement
```

This list contains all key attachements. A key attachement is a definition for a special hotkey.

You can't use the ':' operator (list operator) and you can't use list name completion since there are no names in this list.

```
< l attc <enter>
> Node      Code Qualifier Command
> -----
> 07E681C8   89          0 'wins
> 07E681B0   88          0 'port
> 07E67770   87          0 'devs
> 07E61028   86          0 'libs
> 07E60D30   85          0 'task
> 07E4E728   82          0 'led
> 07E25E08   81          0 'cls
> 07E1F808   80          0 'list
```

```
Node      pointer to key attachement node
Code      key code needed to activate this key attachement
Qualifier  qualifier needed to activate this key attachement
Command   command executed
```

There is no extra information available with the `info` command.

1.8 List Reference : conf

```
Name : 'COnf' |
Cmd  : 'COnf' |
-----+

```

This list contains all autoconfig devices present in the system.

```
< l conf <enter>
> Config Name      : Node      Pri Flags BAddr      BSize      Driver
> -----
> ...

Node      pointer to autoconfig node
Pri       priority
Flags     flags
BAddr     board address
BSize     board size
Driver    pointer to driver

```

The `info` command gives the listing of the config structure.

1.9 List Reference : crsh

```
Name : 'Crsh' |
Cmd  : 'CRsh' |
Some related commands : |
remcrash : remove a crash node |
kill     : kill a task and crash node |
-----+

```

This list contains all crashed tasks. Note that these crashed tasks are still contained in the 'task' list as well.

You can't use the ':' operator (list operator) and you can't use list name completion since there are no names in this list.

```
< l crsh <enter>
> Node      Task      TrapNr  2ndInfo  Guru
> -----
> 07E95DC8 07EBA420 00000005 00000000 0

Node      pointer to crash node
Task      pointer to crashed task

```

```

TrapNr      crash trap number or guru number
2ndInfo     second information (only for guru)
Guru        0 if trap, 1 if guru, 2 if stackoverflow

```

There is no extra information available with the `info` command.

1.10 List Reference : dbug

```

Name : 'DBug'
Cmd  : 'DBug'

Some related commands :

  debug      : control debug nodes
-----+

```

This list contains all debug tasks (tasks you are debugging). Note that these tasks are still contained in the `task` list as well.

```

< l dbug <enter>
> Debug task      : Node      Task      InitPC   TD ID Mode  SMode TMode
> -----+-----
> TTXCalc         : 07EB0C60 07EFBA50 07EF7E80 FF FF NONE  WAIT  NORM

```

```

Node      pointer to crash node
Task      pointer to task
InitPC    initial program counter
TD        task disable counter
ID        interrupt disable counter
Mode      trace mode :
          NONE      not tracing
          TRACE    tracing
          EXEC     executing
SMode     special mode :
          NORM     normal debugging
          CRASH   crashed
          BREAK   breakpoint
          WAIT    waiting for PowerVisor action
          ERROR   error
TMode     trace mode 2 :
          NORM    trace one instruction
          AFTER   trace more instructions
          STEP    trace forever
          UNTIL   trace until address
          REG     trace until register changes
          COND    trace until condition is true
          BRANCH  trace until branch
          FORCE    force tracing (trace f)

```

```

< info dbug:ttxcalc dbug <enter>
> Debug task      : Node      Task      InitPC   TD ID Mode  SMode TMode

```

```

> -----
> TTXCalc          : 07EB0C60 07EFBA50 07EF7E80 FF FF NONE  WAIT  NORM
>
> Node      Number Where      UsageCnt Type Condition
> -----

```

You get a list of all breakpoints :

```

Node      pointer to breakpoint node
Number    breakpoint number
Where     address for breakpoint
UsageCnt  how many times has this breakpoint been passed
Type      breakpoint type :
          T      temporary breakpoint
          N      normal breakpoint
          P      profile breakpoint
          C      conditional breakpoint
          A      countdown breakpoint

```

1.11 List Reference : devs

```

Name : 'Devs' |
Cmd  : 'DEvs' |
-----+

```

This list contains all Exec devices currently in memory.

The information is the same as for the
 libs
 list.

1.12 List Reference : dosd

```

Name : 'DOsd' |
Cmd  : 'DOsd' |
-----+

```

This list contains all dos devices.

```

> Dos device name      : Address  Type      Task      Lock      LockList DiskType
> -----
> includes             : 07E25C4C 00000001 07E18A0C 07E25BD8 00000000 00000000
> docs                 : 07E3CFA4 00000001 07E18A0C 07E3A6C0 00000000 00000000
> auto                 : 07E3A01C 00000001 07E18A0C 07E39FF8 00000000 00000000
> fd                   : 07E33FA4 00000001 07E18A0C 07E33720 00000000 00000000
> rexx                 : 07E403BC 00000001 07E18A0C 07E52078 00000000 00000000
> QUAD                 : 07E51C1C 00000001 07E23C54 07E51BF4 00000000 00000000
> ...

```

```

> PAR          : 07E02ACC 00000000 00000000 00000000 00000004 00000000
> PRT          : 07E02A9C 00000000 00000000 00000000 00000008 00000000
> WB_2.x      : 07E0AD58 00000000 07E10544 00000000 07E0AD88 003E61D4
> DF0         : 07E0EB10 00000000 07E162D4 00000000 07E0EB3C 003E61D4
> Work        : 07E0BB78 00000000 07E18A0C 00000000 07E0AE10 003E61D4

```

```

Address      pointer to dosdevice structure
Type         0 = device
             1 = directory (assign)
             2 = volume
             3 = late-binding assign (AmigaDOS 2.0 only)
             4 = non-binding assign (AmigaDOS 2.0 only)
Task         pointer to task
Lock         pointer to lock (Note this is not a BPTR !)
LockList     pointer to list of outstanding locks
DiskType     disktype

```

There is no extra information available with the `info` command.

1.13 List Reference : exec

```

Name : 'Exec' |
Cmd  : 'Exec' |
-----+

```

This is the listing of the ExecBase structure. You can use the `listaddress` operator to change values in this list. You can't use `info`. There is more information if you have AmigaDOS 2.0.

```

< l exec <enter>
> SoftVer      : 012F      | LowMemChkSum : 0000      | ChkBase      : F81FF833
> ColdCapture  : 00000000 | CoolCapture  : 00000000 | WarmCapture  : 00000000
> SysStkUpper  : 07E02230 | SysStkLower  : 07E00A30 | MaxLocMem    : 00200000
> DebugEntry   : 00F82E88 | DebugData    : 00000000 | AlertData    : 00000000
> MaxExtMem    : 00000000 | ChkSum       : A2BE      | ThisTask     : 07E72728
> IdleCount    : 0004C6D7 | DispCount    : 000292F4 | Quantum      : 0004
> Elapsed      : 0002      | SysFlags     : 0000      | IDNestCnt    : FF
> TDNestCnt    : FE        | AttnFlags    : 0017      | AttnResched  : 0000
> ResModules   : 07E00410 | TaskTrapCode : 07E77696 | TaskExceptCod: 00F83AEC
> TaskExitCode : 00F8242C | TaskSigAlloc : 0000FFFF | TaskTrapAlloc: 8000
> VBlankFreq   : 32        | PowerSupplyFr: 32        | KickTagPtr   : 00000000
> KickChecksum : 00000000 | RamLibPrivate: 07E1E528 | EClockFreq   : 000AD303
> CacheCtrl    : 00002919 | TaskID       : 00000001 | PuddleSize   : 00000000
> MMULock      : 00000000 |

```

1.14 List Reference : fdfi

```

Name : 'FDfi' |
Cmd  : 'FDfi' |

```

```

Some related commands :
|
|
loadfd      : load a fd-file |
unloadfd    : unload a fd-file |
|
-----+

```

This list contains all fd-files loaded by PowerVisor. A fd-file is a collection of library functions. You can load fd-files with `loadfd` command and unload them with the `unloadfd` command.

```

< loadfd libs:exec fd:exec_lib.fd <enter>
> New functions: 0000007E,126

```

```

< l fdfi <enter>
> Library name      : Node      Library  Funcs
> -----
> exec.library      : 07E73690 07E007CC   126

```

Node pointer to fd-file node
Library pointer to corresponding library
Funcs number of functions loaded

```

< info fdfi:exec fdfi <enter>
> Library name      : Node      Library  Funcs
> -----
> exec.library      : 07E73690 07E007CC   126
>
> Supervisor
> ExitIntr
> Schedule
> Reschedule
> Switch
> Dispatch
> Exception
> InitCode
> InitStruct
> ...
> ColdReboot
> StackSwap
> ChildFree
> ChildOrphan
> ChildStatus
> ChildWait

```

You get the list of all functions in this fd-file node.

1.15 List Reference : fils

```

Name : 'Fils' |
Cmd  : 'FILS' |
|
-----+

```

This list contains all open files. This list is in fact a subset of the

```
lock
list. It contains all locks with a size different from 0. This
means that empty files are not present in the file list.
```

You cannot use the ':' operator (list operator) and you can't use list name completion.

```
< l files <enter>
> FileName                : Lock      Access      Size      Key
> -----
> Ram Disk:test           : 07E73FD4  WRITE      154 132730236

Lock      pointer to lock
Access    access type (READ or WRITE)
Size      filesize
Key       disk key
```

There is no extra information available with the info command.

1.16 List Reference : font

```
Name : 'Font' |
Cmd  : 'Font' |
-----+

```

This list contains all fonts currently in memory.

```
< l font <enter>
> Font node name          : Node      Pri YSize XSize Style LoChar HiChar
> -----
> topaz.font              : 07E083F0 0A      8    8    0    32   255
> topaz.font              : 07E083B8 00      9   10    8    32   255
> courier.font            : 07E3223A 00     13    7    0    32   255

Node      pointer to the font node
Pri       priority
YSize     height for font
XSize     width for font
Style     style
LoChar    lowest character defined in this font
HiChar    highest character defined in this font
```

```
< info font:topaz font <enter>
> Font node name          : Node      Pri YSize XSize Style LoChar HiChar
> -----
> topaz.font              : 07E083F0 0A      8    8    0    32   255
>
> Flags                   : 41          | Baseline      : 0006          | BoldSmear     : 0001
> Accessors               : 0014        | CharData     : 00FC25FA     | Modulo        : 00C0
> CharLoc                 : 00FC2276   | CharSpace    : 00000000     | CharKern      : 00000000
```


The font structure is dumped.

1.17 List Reference : func

```
Name : 'FUnc'
Cmd  : 'FUnc'

Some related commands :

    addfunc      : add a function monitor node
    remfunc      : remove a function monitor node
```

This list contains all function monitors for PowerVisor. Each node in this list is created by the `addfunc` command. A function monitor node is a structure containing information about a library function that is being monitored by PowerVisor.

```
< addfunc putmsg led <enter>
```

```
< l func <enter>
```

```
> Function monitor      : Node      Library  Offset Traptask      Count Type
> -----
> putmsg                : 07E94CB0 07E007CC      366 00000000          0 LED
```

```
Node      pointer to node
Library    pointer to library for function
Offset     offset for function in library
Traptask   if not 0 this is the task that is monitored
Count      how many times has this function been called
Type       LED    = only blink led when called
           NORM   = store more information in node
           FULL   = remember full information
           FLED   = full information and powered
           EXEC   = execute command
           SCRA   = scratch registers
```

Use `info` for more information (The information you get is dependant on the type of the function monitor node).

```
< remfunc putmsg <enter>
```

```
< addfunc putmsg <enter>
```

```
< info func:putmsg func <enter>
```

```
> Function monitor      : Node      Library  Offset Traptask      Count Type
> -----
> putmsg                : 07E94E00 07E007CC      366 00000000          228 NORM
>
> input.device          : 07E08B22 14 07E09B28      4096 Wait           TASK -
> RAM                   : 07E23BF8 0A 07E23EE6      1200 Wait           PROC -
> input.device          : 07E08B22 14 07E09B28      4096 Wait           TASK -
```

```

> RAM          : 07E23BF8 0A 07E23EE6      1200 Wait          PROC -
> RAM          : 07E23BF8 0A 07E23EE6      1200 Wait          PROC -
> RAM          : 07E23BF8 0A 07E23EE6      1200 Wait          PROC -
> RAM          : 07E23BF8 0A 07E23EE6      1200 Wait          PROC -
> RAM          : 07E23BF8 0A 07E23EE6      1200 Wait          PROC -

```

You get the 8 last tasks execting 'PutMsg'

```
< remfunc putmsg <enter>
```

1.18 List Reference : graf

```

Name : 'Graf' |
Cmd  : 'GRaf' |
-----+

```

This is the listing of the GraphicsBase structure. You can use the listaddress operator to change values in this list. You can't use info . There is more information if you have AmigaDOS 2.0.

```

< l graf <enter>
> ActiView      : 07E0BEEA | copinit      : 00000420 | cia          : 07E00278
> blitter       : 00000000 | LOFlst       : 00068760 | SHFlst       : 000687F8
> blthd         : 00000000 | blttl        : 07E02598 | bsblthd      : 00000000
> bsblttl       : 07E0269C | vbsrv        : 07E036D2 | timsrv       : 07E036E8
> bltsrv        : 07E036FE | TextFonts    : 07E03714 | DefaultFont  : 07E083F0
> Modes         : 5004      | VBlank       : 00          | gb_Debug     : 00
> BeamSync      : 0000      | sys_bplcon   : 0200      | SpriteResered: 01
> bytereserved  : 00          | Flags        : 0000      | BlitLock     : FFFF
> BlitNest      : FFFF      | BlitWaitQ    : 07E03736 | BlitOwner    : 00000000
> TOF_WaitQ     : 07E03748 | DisplayFlags : 0009      | SimpleSprite : 07E003A8
> MaxDispRow    : 0105      | MaxDispCol   : 01C7      | NormalDispRow: 00DD
> NormalDispCol: 02B4      | NormalDPMX   : 04CA      | NormalDPMY   : 0513
> LastChanceMem: 07E03C30 | LCMptr       : 00000E30 | MicrosPLine  : 3F9D
> MinDispCol    : 005D      | ChipRevBits  : 03          | monitor_id   : 0001
> hedley_count  : 0000      | hedley_flags : 0000      | hedley_tmp    : 0000
> hash_table    : 07E03830 | cur_tot_rows : 0106      | cur_tot_cclks: 00E2
> hedley        : 07E0377C | hedley_sprite: 07E0379C | hedley_sprite: 07E037BC
> hedley_hint   : 00          | hedley_hint2 : 00          | MonitorList  : 07E03808
> a2024_sync_ra: 0000      | ctrl_delta_pa: 0000      | ctrl_delta_nt: 0000
> current_monit: 07E22A68 | default_monit: 07E22A68 | MonListSemaph: 07E082E8
> DispInfoDBase: 07E03C60 | ActiViewCprSe: 07E003C8 |

```

1.19 List Reference : ihan

```

Name : 'IHan' |
Cmd  : 'IHan' |
-----+
Some related commands :

```

```

remihan      : remove an input handler
-----+

```

This list contains all input handlers.

```

< l ihan <enter>
> InputHandler Name   : Node      Pri Data      Code
> -----+-----
> * Blank_Handler    : 07E6DC24 4B  00000000 07E6FD2E
> PowerSnap 1.0 by Nic: 07E62E20 37  00000000 07E62522
> PowerVisor.input    : 07EBDDA6 35  00000000 07EBD2A6
>                     : 07E8135A 33  00000000 07E9955E
> intuition.library   : 07E0C636 32  07E0BEC8 00FEAEE0
> console.device      : 07E0D8DA 00  07E0D7D0 07E0D7A6

```

```

Node      pointer to input handler node
Pri       priority for node (intuition has 50)
Data      pointer to data
Code      pointer to code

```

The 'PowerVisor.input' handler is for PowerVisor and is always there.

There is no extra information available with the `info` command.

1.20 List Reference : intb

```

Name : 'Intb'
Cmd  : 'INTb'
-----+

```

This is the listing of the IntuitionBase structure. You can use the `listaddress` operator to change values in this list. You can't use `info`.

```

< l intb <enter>
> ActiveWindow : 07E5F758 | ActiveScreen : 07E5DB38 | FirstScreen : 07E5DB38
> Flags        : 00056004 | MouseY       : 010A       | MouseX      : 0278
> Seconds      : 18B26983 | Micros      : 0007A120 |

```

1.21 List Reference : intr

```

Name : 'INTR'
Cmd  : 'INTR'
-----+

```

This is the list of all interrupts in the system.

(no examples since it is very difficult to get anything in this list at all).

```

Node      pointer to the interrupt node
Pri       priority
Data      pointer to data
Code      pointer to code

```

1.22 List Reference : libs

```

Name : 'Libs' |
Cmd  : 'LIBS' |
-----+

```

This list contains all libraries currently in memory.

```

< l libs <enter>
> Library node name  : Node      Pri NegSize PosSize Sum      OpenCnt
> -----
> utility.library   : 07E00154 00      204     44 77D20000    12
> ...
> expansion.library : 00000A34 EC      164     390 9CED0000     2
> exec.library      : 07E007CC 9C      780     612 815E0000     2

```

```

Node      pointer to the library node
Pri       priority
NegSize   negative library size
PosSize   positive library size
Sum       checksum for library
OpenCnt   usage count

```

```

< info libs:exec libs <enter>
> Library node name  : Node      Pri NegSize PosSize Sum      OpenCnt
> -----
> exec.library      : 07E007CC 9C      780     612 815E0000     2
>
> IDString          : exec 36.154 (11.12.90)
> Vers              : 0024      | Rev          : 009A      |

```

A dump of the library structure is taken.

1.23 List Reference : lock

```

Name : 'Lcck' |
Cmd  : 'Lcck' |
Some related commands : |
unlock      : unlock a lock |

```

```

|
-----+

```

This list contains all locks.

You cannot use the ':' operator (list operator) and you can't use list name completion.

```

< l lock <enter>
> FileName          : Lock      Access      Size      Key
> -----
> Ram Disk:test     : 07E73FD4  WRITE      154 132730236
> Ram Disk:env/     : 07E21ED4  READ       0 132269196

```

```

Lock      pointer to lock
Access    access type (READ or WRITE)
Size      filesize
Key       disk key

```

There is no extra information available with the info command.

1.24 List Reference : lwin

```

Name : 'LWin'
Cmd  : 'LWin'

Some related commands :

xwin      : open/close 'Extra' logical window
rwin      : open/close 'Refresh' logical window
dwin      : open/close 'Debug' logical window
awin      : open/close 'Rexx' logical window
owin      : open/close 'PPrint' logical window
openlw    : open logical window
closelw   : close logical window

```

This list contains all logical windows for PowerVisor.

```

< l lwin <enter>
> Logical Window    : Node      PWin      width height col  row  viscol visrow
> -----
> Main              : 07EAF9F0 07EAF6A8  86    51    0   50    0     0

```

```

Node      pointer to logical window node
PWin      pointer to physical window
width     visible width of logical window (in characters)
height    visible height of logical window (in characters)
col       current column position
row       current row position
viscol    first visible column
visrow    first visible row

```

```

< info lwin:main lwin <enter>
> Logical Window      : Node      PWin      width height col  row  viscol visrow
> -----
> Main                : 07EAF9F0 07EAF6A8      86      51      0   50      0      0
>
> Box                 : 07EAF710 | rx          : 0000      | ry          : 000B
> rw                  : 02B4      | rh          : 019B      | Flags       : 00000007
> TA                  : 07EAF1A1 | Font       : 07E05C18 | ocol        : 0056
> orow                : 0033      | NumLines   : 0033      | NumColumns  : 0056
> Buffer              : 07EAF5A8 | File       : 07E9A384 | LinesPassed : 0009
> Active              : 01          | TopBorder  : 0B        | rtop        : 0000

```

1.25 List Reference : memr

```

Name : 'Memr' |
Cmd  : 'MEMR' |
-----+

```

This list contains all available memory.

```

< l memr <enter>
> Memory node name   : Node      Pri  Attr First      Lower      Upper      Free
> -----
> expansion memory   : 07E00000 1E    261 07E88760 07E00020 07F80000 617384
> chip memory        : 00000400 F6     771 00000988 00000420 00200000 1669432

```

```

Node      pointer to the memory node
Pri       priority of the memory node
Attr     attributes for that memory
First    first free memory
Lower    lowest possible free memory
Upper    highest possible free memory
Free     total free memory

```

```

< info memr:chip memr <enter>
> Memory node name   : Node      Pri  Attr First      Lower      Upper      Free
> -----
> chip memory        : 00000400 F6     771 00000988 00000420 00200000 1669432
>
> 00000988          8
> 00000D50          16
> 00006E90          8
> 00033080          16
> 000345A0          40
> 00068540         368
> 00068890        1668976

```

info shows all the free blocks. You can examine memory fragmentation with this list.

The first argument in the list is the address of the free block. The second argument is the size (decimal).

1.26 List Reference : moni

```

|
Name : 'MONi' |
Cmd  : 'MONi' |
|
-----+

```

This list contains all monitors present in the system (AmigaDOS 2.0 only).

```

< l moni <enter>
> Monitor node name   : Node      Pri SubSys SubType Library  Init
> -----
> multiscan.monitor  : 07E22BE8 00    2    4    07E03688 00FCE654
> ntsc.monitor       : 07E22A68 00    2    4    07E03688 00FCE654
> pal.monitor        : 07E08318 00    2    4    07E03688 00FCE654

Node      pointer to monitor node
Pri       priority
SubSys    sub system number
SubType   subtype
Library   library for monitor
Init      init routine

```

```

< info moni:pal moni <enter>
> Monitor node name   : Node      Pri SubSys SubType Library  Init
> -----
> pal.monitor        : 07E08318 00    2    4    07E03688 00FCE654
>
> Flags              : 0002      | ratioh          : 00000010 | ratiov          : 00000010
> tot_rows           : 0138      | tot_colorcloc: 00E2      | DeniseMaxDisp: 01C7
> BeamCon0           : 0020      | min_row         : 001D      | Special         : 00000000
> OpenCount          : 0001      | transform       : 00FCE662 | translate       : 00FCE676
> scale              : 00FCE69E | xoffset         : 0009      | yoffset         : 0000
> LegalView          : 07E0835A | maxoscan       : 00FCE6A2 | videoscan      : 00FCE6B4
> DeniseMinDisp     : 005D      | DispCompatibl  : 0000      | DispInfoDBase : 07E0836E
> DIDBSemaphore     : 07E0837C |

```

You get the listing of the monitor structure.

1.27 List Reference : port

```

|
Name : 'Port' |
Cmd  : 'Port' |
|
-----+

```

This list contains all named message ports currently in the system.

```

< l port <enter>
> MsgPort node name   : Node      Pri SigBit SigTask
> -----
> REXX                : 07E459DC 00          31 07E4DD38

```

```

> AREXX                : 07E4E5C8 00      30 07E4DD38
> AddTools by Steve Ti: 07E41990 00      31 07E5B248
> PowerVisor-port     : 07E77F7A 00        1 00000000
> REXX_POWERVISOR     : 07E605A8 00      24 07E72728
> * Blank_Port        : 07EB7348 00      30 07E605D0
> IPrefs.rendezvous   : 07E227F0 E2      31 07E28330
> SetPatch Port       : 07E227C0 9C        0 00000000

```

```

Node      pointer to the message port node
Pri       message port priority
SigBit    signal bit
SigTask   task to signal

```

There is no extra information available with the `info` command.

The 'REXX_POWERVISOR' port and the 'PowerVisor-port' are used by PowerVisor.

1.28 List Reference : pubs

```

Name : 'PUBs' |
Cmd  : 'PUBs' |
-----+

```

This list contains all public screens present in the system (AmigaDOS 2.0 only).

```

< l pubs <enter>
> PubScreen node name : Node      Pri Screen  Visitors SigTask  SigBit
> -----
> Workbench           : 07E23970 00  07E2D258    4    00000000   255

Node      pointer to public screen node
Pri       priority
Screen    pointer to Intuition screen
Visitors  number of visitor windows on public screen
SigTask   task to signal when screen closes
SigBit    signal bit to use

```

```

< info pubs:workbench pubs <enter>
> PubScreen node name : Node      Pri Screen  Visitors SigTask  SigBit
> -----
> Workbench           : 07E23970 00  07E2D258    4    00000000   255
>
> Flags              : 0000      | Size              : 0028      |

```

Some extra variables from public screen structure.

1.29 List Reference : pwin


```

Name : 'PWin'
Cmd  : 'PWin'

Some related commands :

  openpw      : open physical window
  closepw     : close physical window
-----+

```

This list contains all physical windows for PowerVisor.

```

< l pwin <enter>
> Physical Window      : Node      Window  Code Qualifier
> -----
> Main                 : 07EAF6A8 07EAF740 0000 0000

Node      pointer to physical window node
Window    pointer to Intuition window
Code      keycode
Qualifier keyqualifier

```

```

< info pwin:main pwin <enter>
> Physical Window      : Node      Window  Code Qualifier
> -----
> Main                 : 07EAF6A8 07EAF740 0000 0000
>
> NewWindow           : 07EAF6B6 | SigSet           : 04000000 | BorderLeft   : 00
> BorderTop           : 00         | BorderRight      : 00         | BorderBottom : 08
> Box                 : 07EAF710 | Global           : 07E4BD58 | LWList       : 07EAF6FE

```

1.30 List Reference : resm

```

Name : 'RESM'
Cmd  : 'RESm'

Some related commands :

  remres      : remove a resident module
-----+

```

This list contains all resident modules.

```

< l resm <enter>
> Resident name       : Address  Pri Version Flags IDString
> -----
> expansion.library   : 00F83D78 6E      36 02      expansion 36.96 (11.12.90)
> exec.library        : 00F800BA 69      36 02      exec 36.154 (11.12.90)
> diag init          : 00F83D92 69      36 01      diag init
> utility.library     : 00FBB03A 67      36 81      utility 36.77 (7.12.90)

```

```

> potgo.resource      : 00FAB7FC 64          36 81          potgo 36.19 (9.4.90)
> cia.resource       : 00F88ACC 50          36 01          cia 36.31 (31.7.90)
> ...
> workbench.task     : 00FBBA1A 88          36 00          Pre-2.0 LoadWB stub
> workbench.library  : 00FEC970 88          36 80          wb 36.2720 (11.12.90)
> con-handler        : 00F88DC2 87          36 00          con-handler 36.62 (3.12.90)
> shell              : 00FB2C3C 86          36 00          shell 36.114 (7.12.90)
> ram-handler        : 00FAB998 85          36 00          ram 36.37 (7.12.90)

```

```

Address    pointer to resident module structure
Pri        priority
Version    version
Flags      some flags
IDString   ID string

```

There is no extra information available with the `info` command.

1.31 List Reference : reso

```

Name : 'Reso' |
Cmd  : 'RESO' |
-----+

```

This list contains all Exec resources currently in memory.

The information is the same as for the
 libs
 list.

1.32 List Reference : scrs

```

Name : 'Scrs' |
Cmd  : 'SCrs' |
Some related commands : |
    closescreen : close a screen |
-----+

```

This list contains all screens. There is more information in AmigaDOS 2.0.

```

< l scrs <enter>
> Screen name      : Address  Left  Top Width Height FirstWindow
> -----
> PowerVisor      (V1.00/: 07E5DB38   0   0  692   442 07E5F758
> Workbench Screen : 07E280D0   0 -582 692  1024 07E10348

```

```

Address    pointer to the screen structure

```

```

Left      left coordinate for screen (always 0 in AmigaDOS 1.3)
Top       top coordinate for screen
Width     width for screen
Height    height for screen
FirstWindow pointer to the first window on this screen

```

```

< info scrs:powervisor scrs <enter>
> Screen name      : Address  Left  Top Width Height FirstWindow
> -----
> PowerVisor      (V1.00/: 07E5DB38   0   0   692   442 07E5F758
>
> Flags           : 021F           | Font           : 07E7B804 | ViewPort      : 07E5DB64
> RastPort        : 07E5DB8C | BitMap        : 07E5DBF0 | FirstGadget   : 07E58204
> DefaultTitle    : PowerVisor (V1.10 beta, AmigaDOS 2.0) © J.Tyberghein
> DetailPen       : 00           | BlockPen      : 01           | ExtData       : 00000000
> UserData        : 00000000 | BarHeight     : 0A           | BarVBorder    : 01
> BarHBorder      : 05           | MenuVBorder   : 02           | MenuHBorder   : 04
> WBorTop         : 02           | WBorLeft     : 04           | WBorRight    : 04
> WBorBottom     : 02           | LayerInfo     : 07E5DC18 | BarLayer      : 07E5F668
>
> Flags: CUSTOMSCREEN SHOWTITLE SCREENHIRES

```

The complete screen structure is printed.

1.33 List Reference : sema

```

Name : 'SEma' |
Cmd  : 'SEMa' |
-----+

```

This is the list of all semaphores in the system.

(no examples since it is very difficult to get anything in this list at all).

```

Node      pointer to the semaphore node
Pri       priority
NestCount nest count
QueueCount queue count
Owner     owner of semaphore

```

1.34 List Reference : stru

```

Name : 'STru' |
Cmd  : 'STru' |
-----+
Some related commands :
addstruct : add structure definition nodes

```

```

remstruct   : remove a structure definition node   |
clearstruct : remove all structure definition nodes |
                                                    |
-----+

```

This list contains all structure definitions loaded by PowerVisor. You can load structure definitions with the `addstruct` command, and you can remove them with the `remstruct` or `clearstruct` commands.

```

< l stru <enter>
> Struct node name      : Node      Pri InfoBlock Strings  Length
> -----
> IS                    : 07E95640 FD  07EB020A 07E66CE2    22
> IV                    : 07EB0228 FD  07EB024A 07E8A902    12
> IO                    : 07EB0270 FD  07EB0292 07EB02CA    32
> LH                    : 07EB04B0 FD  07EB0CE2 07EB05C2    14
> ...
> TC                    : 07EB1548 FD  07EB156A 07EB160A    84
> LIB                   : 07EB0478 FC  07EB0C42 07EB0C9A    34
> MLH                   : 07EB0D18 FC  07EB0D3A 07E8BE72    12
> ETask                 : 07EB16A8 FA  07EB16CA 07EB171A    86
> StackSwapStruct      : 07EB1768 F0  07EB17A2 07EB17CA    12

```

```

Node          pointer to structure definition node
Pri           priority (internal use only)
InfoBlock     pointer to infoblock (internal use only)
Strings       pointer to all strings in structure
Length        size of structure represented by structure definition

```

There is no extra information available with the `info` command.

1.35 List Reference : task

```

Name : 'Task'
Cmd  : 'Task'

Some related commands :

kill       : kill a task or process
freeze    : freeze a task or process
unfreeze  : unfreeze a task or process
-----+

```

This list contains all tasks and processes. Frozen tasks are also in this list. There is more information if you have AmigaDOS 2.0.

```

< l task <enter>
> Task node name      : Node      Pri StackPtr  StackS Stat Command      Acc
> -----
> Background Process : 07E52140 00  07E57EA8   4096 Rdy  clock      (04) -
> ...
> RAM                 : 07E23BF8 0A  07E23EE6   1200 Wait                PROC -
> input.device       : 07E08B22 14  07E09B28   4096 Wait                TASK -

```

```
> Background Process : 07E72728 04 07E882AA 12000 Run pv (05) -

Node      pointer to the task or process
Pri       priority
StackPtr  stack pointer
StackS    stack size
Stat      state of task: Rdy (Ready), Run (Running), Wait (Waiting),
           Cold (Frozen with 'freeze')
Command   command executing (only for Cli processes)
           (xx) is cli number
           PROC is process
           TASK is task
Acc       is accounting information. '-' means no accounting done
```

```
< info 07E52480 task <enter>
```

```
> Task node name      : Node      Pri StackPtr  StackS Stat Command      Acc
> -----
> Background CLI     : 07E52480 04 07E8886E 12000 Run  pv          (01) -
>
> rc2                : 00000000 | rc                : 00000000 | CmdDir      : 07E1EFB4
> StdIn              : 07E1E3FC | StdOut            : 07E1EF84 | CurIn       : 07E1E3FC
> CurOut              : 07E1EF84 | Backgrnd         : FFFFFFFF | Interactive  : 00000000
> DefStack           : 00000BB8 | FailLevel        : 0000000A | Module      : 07EDEC2C
> SetName            : Work:Macro
> Prompt             : %N.%S>
> CmdFile            :
>
> SegList            : 07E1E63C | StackSize        : 00000C80 | TaskNum     : 00000001
> StackBase          : 07E52564 | rc2              : 00000000 | CurDir      : 07E1EF60
> CIS                : 07E1E3FC | COS              : 07E1EF84 | ConsoleTask : 07E6830C
> FileSystemTas     : 07E0E824 | CLI              : 07E1E484 | ReturnAddr  : 07E88BC0
> PktWait            : 00000000 | WindowPtr        : 00000000 | HomeDir     : 07E1FAA0
> Flags              : 00000004 | ExitCode         : 00000000 | ExitData    : 00000000
> LocalVars          : 07E52550 | ShellPrivate     : 00000000 | CES         : 00000000
> Arguments         :
>
> IDNestCnt          : FF          | TDNestCnt        : 01          | SigAlloc    : FFE0FFFF
> SigWait            : 00000100 | SigRecvd         : 04000000 | SigExcept   : 00000000
> TrapAlloc          : 8000          | TrapAble         : 0000        | ExceptData  : 00000000
> ExceptCode         : 00F83A04 | TrapData         : 00000000 | TrapCode    : 07EE3884
> SpLower            : 07E85CE8 | SpUpper          : 07E88BC8 | SpReg       : 07E889D6
> MemEntry           : 07E524CA | UserData         : 00000000 |
```

The first shown structure is the Cli structure (this structure is not shown if the task is not a Cli).

The second structure is the process structure (only for processes).

The last structure is the task structure. The task structure is always shown.

1.36 List Reference : wins

```

|
Name : 'Wins' |
Cmd : 'Wins' |
|
Some related commands : |
|
closewindow : close a window |
|
-----+

```

This list contains all windows for all screens. There is more information in AmigaDOS 2.0.

```
< l wins <enter>
```

```

> Window name      : Address  Left  Top Width Height WScreen
> -----
>                  : 07E5F758   0   12   692   430 07E5DB38
>                  : 07E8BC70   0    0   704   456 07E88CE0
> My Shell         : 07E10348   0  568   692   456 07E280D0
> Clock           : 07E41788  558  336   120   140 07E280D0
>                  : 07E3B410   0   16   692  1008 07E280D0

```

```

Address      the address of the window structure
Left         the left coordinate in the screen
Top          the top coordinate in the screen
Width        the window width
Height       the window height
WScreen      the screen for the window

```

```
< info 07E5F758 wins <enter>
```

```

> Window name      : Address  Left  Top Width Height WScreen
> -----
>                  : 07E5F758   0   12   692   430 07E5DB38
>
> MinWidth        : 0064      | MinHeight       : 0064      | MaxWidth        : FFFF
> MaxHeight       : FFFF      | Flags           : 04033900 | MenuStrip       : 00000000
> ScreenTitle     : PowerVisor (V1.10 beta, AmigaDOS 2.0) © J.Tyberghein
> FirstReques    : 00000000 | DMRequest       : 00000000 | ReqCount        : 0000
> RPort          : 07E5F800 | Pointer         : 00000000 | PtrHeight       : 00
> PtrWidth        : 00        | XOffset         : 00        | YOffset         : 00
> IDCMPFlags     : 004C0062 | UserPort        : 07E5F988 | WindowPort      : 07E5F960
> MessageKey     : 07E71E48 | DetailPen       : 00        | BlockPen        : 00
> CheckMark      : 07E0B960 | ExtData         : 00000000 | UserData        : 00000000
> BorderLeft     : 00        | BorderTop       : 00        | BorderRight     : 00
> BorderBottom   : 00        | BorderRPort     : 00000000 | Parent          : 07E8BC70
> Descendant     : 07E41788 | GZZMouseX       : 00C8      | GZZMouseY       : 0172
> GZZWidth       : 02B4      | GZZHeight       : 01AE      | IFont           : 07E083F0
> MoreFlags      : 00000000 |
>
> Flags: SMARTREFRESH BACKDROP BORDERLESS ACTIVATE WINDOWACTIVE RMBTRAP
> NOCAREREFRESH WINDOWTICKED
> IDCMP: NEWSIZE GADGETDOWN GADGETUP ACTIVEWINDOW INACTIVEWINDOW INTUITICKS

```

This is the complete window structure. The 'Flags' and 'IDCMP_Flags' are printed with their actual bit-defined values.