NAME

sgen – system generation program

SYNOPSIS

sgen [-uv] spec [file]

DESCRIPTION

Sgen is a program which builds a core image of the basic *MERT* operating system. The functions performed are:

1) It generates the low core image, allocating interrupt vectors and establishing the linkage by which processes can attach to the interrupts at run time.
2) Relocates the kernel text and data segments so that they each start at 20000(8).
3) Appends the basic modules needed at boot to the kernel core image. These include:
    1) The system library
    2) The process for the root device
    3) The process for the swap device
    4) The file manager process
    5) The process manager bootstrap process
    6) The nub process (a subtask of the process manager)
    7) System initialization process
4) Generates a table of pathnames of processes to be created by the process manager at boot time.

Input to *sgen* consists of flags and a specification file. The flags are:

u Do an update sysgen, that is do not regenerate a new low core image.
v Verbose mode - print out a map of memory at the end of system generation.

The specification file consists of lines containing a keyword and a parameter list. The elements in the parameter list may be separated by blanks, tabs, or commas. A comment delimited by /* may be added to any line. All numerical parameters are assumed to be octal unless terminated by a decimal point. The keywords are:

fmgr pathname
The file *pathname* contains the process file (the output of *ldp)* of the file manager process

init pathname
The file *pathname* contains the relocatable initialization process.

kernel pathname
The file *pathname* contains the relocatable kernel.

lowcore pathname
The file *pathname* contains the relocatable binary of the lowcore module. This module is generated by *sgen* from the assembly language files *lcor0.s* and *lcor1.s.*

nubprc pathname
The file *pathname* contains the process file (output of *ldp)* of the process which creates supervisor mode processes. This process is actually a subtask of the process manager. It is included in system generation to simplify booting.

pmboot pathname
The file *pathname* contains the process file (the output of *ldp)* of the process which will create the process manager.

pmgr pathname

The file *pathname* contains the process file of the process manager. *Sgen* simply passes this pathname to the init process which in turn passes it to pmboot. The pathname must start from the root and the file must exist on the root file system at boot time.

rootdev major minor

Major is the major device number of the root file system. *Minor* is the minor device number of the root file system.

rootprc pathname

The file *pathname* is the process file of the process which services the device containing the root file system.

swapdev major minor

Major is the major device number of the swap device. *Minor* is the minor device number of the swap device.

swapprc pathname

The file *pathname* is the process file of the process which services the device which contains the system swap area.

syslib pathname

The file *pathname* contains the public library file (output of *ll)* set up to execute in kernel base register six. If this keyword is excluded, no system library will be generated. By convention the pathname of the system library file is /mrt/syslib.

user pathname

The file *pathname* is the process file of a process to be started up by the kernel initialization process at boot time. Only one pathname can be specified with each *user* keyword. The *user* keyword may be repeated nine times.

The following key words are associated with construction of low core and memory management tables. These specifications are ignored if the "-u" option is specified.

memory start size [start size [start size]]

Physical memory is broken up into one, two, or three partitions. *Start* is the beginning (64 byte) block address of the partition, *size* is the number of (64 byte) blocks in the partition. At system initialization time the actual size of memory is determined and the size of the last partition is adjusted to reflect the top of memory.

messages n

N - 16 word message buffers are allocated. The default is 32 and the maximum allowed is 160.

nrsde n

N resident segment descriptor entries (RSDE) will be allocated in the kernel private data segment (not low core). The number of RSDEs determines the maximum number of segments that can exist in the system at any time. One should allow about 3.7 segments per process, but not more than a total of 500. The default is 250.

processes n

N process table entries (DCT) are allocated. The default is 50 and the maximum is 127.

stack size

The size of the system stack is *size* bytes. It must be at least 64 bytes and not more than 8190 bytes. The system stack resides at the high address end of the lowcore segment.

v addr csr

the *v* (for vector) keyword is included for defining non-standard devices or standard devices which use interrupt vectors and/or control and status registers which do not conform with DEC conventions. *Addr* is the interrupt vector address used by the device and *csr* is the address of the device control and status register (the register containing the interrupt enable bit).

ports n

N port table entries are created. The minimum number is 2 and the maximum is 16. The default is 4.

nsde n

This determines the size of the SDE table that will be allocated in the kernel private data segment (not low core). The maximum for n is 250; default is 100 entries.

The following keywords are provided to handle standard DEC devices and are simply "built in" *v* keywords (e.g. the program knows the vector and csr addresses).

```
console  pc11    kw11p   parity  plot
ad01     afc11   aa11d   aa11l   lp11
ls11     rf11    rc11    tc11    tm11
tu11     ht11    rk11    cd11    cm11
cr11     udc11   rp11    rp03    hp11
rjp04    tf11    ta11    dc11    kl11
dl11a    dl11b   dl11c   dl11d   dl11e
dm11a    dm11b   dn11    dp11    dr11a
dr11c    dt11    dj11    dh11    dq11
du11
```

FILES

/bin/ld
/bin/as
lcor0.s generated by sgen and used to form lowcore image
lcor1.s contains constants which must be in I = D = phyical memory

ALSO SEE

ldp(e), ll(e)