NAME

getseg – get user segment

SYNOPSIS

(getseg = 71.; not in assembler)
(function code in r0)
sys getseg; segstruct
(code in r0)

segd = makeseg(segstrp)
segd = getseg(segstrp)
code = rmovseg(segstrp)
segd = connseg(segstrp)
code = discseg(segstrp)
segd = getsnam(segstrp)
struct segstruct *segstrp;

DESCRIPTION

The six routines manipulate user segments in the user's address space (either I or D). The user may have up to six (6) segments in his address space in addition to the code, data and stack segments at any one time. The user can manipulate more than six segments at a time but only six of these may be in his active address space simultaneously.

From assembly language, the function code in register *r0* specifies the request type.

0	A new segment is created in the user's address space.
1	The user may put a segment known by name in his address space.
2	A segment is completely removed from the user's address space and the segment will no longer be one of his available segments.
3	An already existing segment (previously put in his address space by a call to *makeseg* or *getseg*) is put in the user's active address space.
4	A segment is disconnected from the user's address space. This segment is still available to be put in his active address space at a later time.
5	The name of a user segment is determined. The segment may be a part of the user's address space as the result of the inclusion of a public library.

All uses of this system call require the reference to a five-word (5) structure as follows:

```
struct  segstruct {
        int     segname[2];
        char    perm;
        char    breg;
        int     segsize;
        char    *segaddr;
};
```

The segment name *segname* is a 32-bit quantity and must be specified when a segment is being referred to. Segments are normally managed by maintaining a table of the above structures, one for each segment. If the name of a segment is not known, the segment descriptor code, *segd* may be used to identify it. This segment descriptor refers to an entry in the process PCB table of user segments, a value of *segd* equal to zero being the first entry. Thus if the segment name *segname* passed as an argument to the system call is less than 248, it is taken to be a segment descriptor, *segd*. *segd* (i.e. segname[0] == 0 && segname[1] < 248). *Makeseg* creates a segment in the user's address space. The name, *segname* must be zero for a new segment to be created. However, for *getseg*, the name must be non-zero since the segment must already exist. The base register to be used to point to the user segment may be specified by *breg*. A value of *breg* from 0 to 7 will put the segment in the I-mode address space of the user; a value from 8 to 15 will put the segment in the D-mode address space of the user. If the user wishes the system to determine the next available base register for the segment, a value of −1 must be given for *breg*. If the user wishes the system to select a base register starting at a given base register, then it should specify *breg* in the range -8 to -16. A *breg* value of -8 causes the system to start looking at base register 0, a value of -9 starts the search at base register 1, etc. The address of the logical segment will start on a segment boundary, i.e. a multiple of 8192 bytes and will be *segsize* bytes in length. The user must specify the length of the segment to be created; however, if the segment already exists and the user has been passed the name of the segment in a message, a length of zero bytes should be specified to ensure that a brand new segment is not created. This segment may be shared between the parent and any number of its children. The share permissions of the segment *perm* are specified for the child in bits <5-3> and for the parent in bits <2-0> as follows:

<5-3>	0 - not shared
	2 : read only
	6 : read/write
	7 : shareable by process created by "pcreat"
<2-0>	2 : read only
	6 : read/write.

A user segment number (descriptor) is returned in *segd*. The name of the segment is returned in *segname* and may be used for future reference to the segment. The size of the segment in bytes is returned in *segsize*. In all cases the starting address of the segment is returned in *segaddr*. This segment is made one of the user's active segments. The shared segment is maintained across all system *fork's* and *exec's*. All segments are deactivated across an *exec*, so a *connseg* must be done to reactivate them. Synchronization may be achieved by means of the *sendev* and *waitev* primitives.

Getseg is used to get an already existing and named segment into the user's address space.

Rmovseg is used to remove a segment completely from the user's address space. The slot in the user's process PCB is also freed up for later use. All that the user need specify in the structure is the segment name. The name may be either a 32-bit name or a number less than 256, in which case it is treated as a *segd*. The segment can no longer be put back in the user's address space, unless another process still has it as one of its segments and the user invokes *getseg* again with the name of the segment. A positive return code indicates success, i.e. the segment did exist and was in fact removed.

Connseg connects a segment in the process's PCB table into the user's address space at the specified address. If the value of *breg* is −1, the next available user base register will be used, other wise the one specified by *breg* will be used. The name of the segment must be passed in *segname*. If the name is less than 256, it is taken to be the segment descriptor, *segd*. The segment descriptor, *segd* is returned in *r0*. The size of the segment in bytes is returned in *segsize*. The starting address of the user segment is returned in *segaddr*.

Discseg disconnects a user segment from his active address space. The name specified has the same convention as specified by *connseg* above. The segment may be connected as an active segment at a later time. A positive return code indicates a successful function call.

Getsnam returns the name of the segment specified by *segd* in the segment name field, *segname*. The 32-bit name of the segment is returned in *segname*. In addition, the third word of the segment structure does not contain the base register and permissions, rather it contains a copy of the status word of the segment in the PCB. This is useful for debugging. The segment descriptor is returned in *r0*. The segment is not put in the user's active address space.

SEE ALSO

sendev(f), waitev(f).

DIAGNOSTICS

The error bit (c-bit) is set if a new segment cannot be created because of insufficient swap space. From C, a −1 value is returned on an error. Other error conditions possible are the non-existence of the named segment or insufficient address space in the user's address space for the segment.

BUGS

Programs that run in separated I and D space cannot get segments into their I space.