NAME

salloc – string allocation and manipulation

SYNOPSIS

(get size in r0)
jsr pc,allocate
(header address in r1)

(get source header address in r0,
destination header address in r1)
jsr pc,copy

jsr pc,wc

(all following routines assume r1 contains header address)

jsr pc,release

(get character in r0)
jsr pc,putchar

jsr pc,lookchar
(character in r0)

jsr pc,getchar
(character in r0)

(get character in r0)
jsr pc,alterchar

(get position in r0)
jsr pc,seekchar

jsr pc,backspace
(character in r0)

(get word in r0)
jsr pc,putword

jsr pc,lookword
(word in r0)

jsr pc,getword
(word in r0)

(get word in r0)
jsr pc,alterword

jsr pc,backword
(word in r0)

jsr pc,length
(length in r0)

jsr pc,position
(position in r0)

jsr pc,rewind

jsr pc,create

jsr pc,fsfile

jsr pc,zero

DESCRIPTION

This package is a complete set of routines for dealing with almost arbitrary length strings of words and bytes. It lives in */lib/libs.a*. The strings are stored on a disk file, so the sum of their lengths can be considerably larger than the available core. A small buffer cache makes for rea-

sonable speed.

For each string there is a header of four words, namely a write pointer, a read pointer and pointers to the beginning and end of the block containing the string. Initially the read and write pointers point to the beginning of the string. All routines that refer to a string require the header address in r1. Unless the string is destroyed by the call, upon return r1 will point to the same string, although the string may have grown to the extent that it had to be be moved.

Allocate obtains a string of the requested size and returns a pointer to its header in r1.

Release releases a string back to free storage.

Putchar and *putword* write a byte or word respectively into the string and advance the write pointer.

Lookchar and *lookword* read a byte or word respectively from the string but do not advance the read pointer.

Getchar and *getword* read a byte or word respectively from the string and advance the read pointer.

Alterchar and *alterword* write a byte or word respectively into the string where the read pointer is pointing and advance the read pointer.

Backspace and *backword* read the last byte or word written and decrement the write pointer.

All write operations will automatically get a larger block if the current block is exceeded. All read operations return with the error bit set if attempting to read beyond the write pointer.

Seekchar moves the read pointer to the offset specified in r0.

Length returns the current length of the string (beginning pointer to write pointer) in r0.

Position returns the current offset of the read pointer in r0.

Rewind moves the read pointer to the beginning of the string.

Create returns the read and write pointers to the beginning of the string.

Fsfile moves the read pointer to the current position of the write pointer.

Zero zeros the whole string and sets the write pointer to the beginning of the string.

Copy copies the string whose header pointer is in r0 to the string whose header pointer is in r1. Care should be taken in using the copy instruction since r1 will be changed if the contents of the source string is bigger than the destination string.

Wc forces the contents of the internal buffers and the header blocks to be written on disc.

An in-core version of this allocator exists in *dc* (I), and a permanent-file version exists in *form* and *fed* (VI).

FILES

/lib/libs.a	library, accessed by *ld ... -ls*
alloc.d	temporary file for string storage

SEE ALSO

alloc (III)

DIAGNOSTICS

‘error in copy’ – disk write error encountered in *copy.*
‘error in allocator’ – routine called with bad header pointer.
‘cannot open output file’ – temp file *alloc.d* cannot be created or opened.
‘out of space’ – no sufficiently large block or no header is available for a new or growing block.

BUGS