NAME

hcreate, hdestroy, hsearch - hash table management

SYNOPSIS

```
#include <search.h>
ENTRY *hsearch(ENTRY item, ACTION action);
int hcreate(unsigned nel);
void hdestroy(void);
```

DESCRIPTION

These three functions allow the user to create a hash table which associates a key with any data.
First the table must be created with the function hcreate(). nel is an estimate of the number of entries in the table. hcreate() may adjust this value upward to improve the performance of the resulting hash table. The GNU implementation of hsearch() will also enlarge the table if it gets nearly full. malloc(3) is used to allocate space for the table.
The corresponding function hdestroy() frees the memory occupied by the hash table so that a new table can be constructed.
item is of type ENTRY, which is a typedef defined in <search.h> and includes these elements:

```
     typedef struct entry
       {
         char *key;
         char *data;
       } ENTRY;
```

key points to the zero-terminated ASCII string which is the search key. data points to the data associated with that key. (A pointer to a type other than character should be cast to pointer-to-character.) hsearch() searches the hash table for an item with the same key as item, and if successful returns a pointer to it. action determines what hsearch() does after an unsuccessful search. A value of ENTER instructs it to insert the new item, while a value of FIND means to return NULL.

RETURN VALUE

hcreate() returns NULL if the hash table cannot be successfully installed.

hsearch() returns NULL if action is ENTER and there is insufficient memory to expand the hash table, or action is FIND and item cannot be found in the hash table.

EXAMPLE

The following program inserts 24 items in to a hash table, then prints some of them.

```
    #include <stdio.h>
    #include <search.h>

    char *data[]={ "alpha", "bravo", "charley", "delta",
         "echo", "foxtrot", "golf", "hotel", "india", "juliette",
         "kilo", "lima", "mike", "november", "oscar", "papa",
         "quebec", "romeo", "sierra", "tango", "uniform",
         "victor", "whiskey", "x-ray", "yankee", "zulu"
     };

    int main()
    {
      ENTRY e, *ep;
      int i;

      /* start with small table, and let it grow */
      hcreate(3);
      for (i = 0; i < 24; i++)
        {
          e.key = data[i];
          /* data is just an integer, instead of a pointer
             to something */
          e.data = (char *)i;
          ep = hsearch(e, ENTER);
          /* there should be no failures */
          if(ep == NULL) {fprintf(stderr, "entry failed\n"); exit(1);}
        }
      for (i = 22; i < 26; i++)
        /* print two entries from the table, and show that
           two are not in the table */
        {
          e.key = data[i];
          ep = hsearch(e, FIND);
          printf("%9.9s -> %9.9s:%d\n", e.key, ep?ep->key:"NULL",
                 ep?(int)(ep->data):0);
        }
      return 0;
    }
```