C Notebook

Compiled and Linked by Lee Calcraft

RESCUING dbox_fillin() KEYPRESSES

Boris Perryman

When using both writable icons and menu icons in a dialogue box, the functions *dbox_fillin()* and *dbox_fillin_fixedcaret()* do not normally react to keypresses corresponding to the capital letters of the menu icons - such as *Save*, *reDo* etc. because all keypresses are directed to the writable icons.

But if the writable icon with the input focus contains a validation string which excludes these characters, the problem is solved. So for example a validation string of A0-9 would allow a calculator to use the letter "C" as a shorthand for *Calculate* etc. Note that the new "K" option in validation strings in RISC OS 3 specifically caters for this situation.

WATCH OUT FOR BOUNDARIES

David Pilling

When creating structures it is worth noting that they are always rounded up to word boundaries, and that ints will be put on *word* boundaries, chars on byte boundaries, and shorts on two-byte boundaries. So it is better to write:

```
typedef struct fixedbit  {
 int  word;
 char  byte;
 short twobytes;
} fixedbit;
```

than:

```
typedef struct fixedbit  {
 char  byte;
 int  word;
 short twobytes;
} fixedbit;
```

The size of the first is 8 bytes, the second 12 bytes.

USING atexit()

Tony Shew

If you want to ensure that vital things such as saving files are performed before *exit()* is called in a non-Wimp program, you can use the standard ANSI function *atexit()* to register functions to be called immediately prior to termination. See K & R for further details. This also works with Wimp tasks, but here the better approach is to use the Pre-quit message.

MORE EOF ERRORS

Alun Evans

The cure given in April's C Notebook for the cryptic EOF message does not always do the trick. Using Source Edit it seems that some invisible character such as a tab may become inserted before the end of the file. I find that the following rigmarole fixes the problem. Use Ctrl-Down to move the caret to the true end of the file, then mark from the final closing brace of your code to the end of the file, and use *delete block* to delete the marked area. Finally insert a final carriage return after the closing brace.

A STAND-ALONE DEBUGGING FUNCTION

David Pilling

The following function can form a simple but effective debugger. It prints debugging information on a specified line (given as the first parameter of the call) on the Desktop, overwriting whatever happens to be displayed there.

Like last month's debugging function it can be passed a variable number of arguments exactly as with *printf()*. Note that its use of *akbd_pollsh()* enables you to pause the program each time that it encounters a call to the function - just hold down Shift, then press any key to continue.

Here is an example of its use:

```
dprintf(0,"x=%d name=%s",x,name);
```

This would display the two variables on the top text line of the screen.

```
#include "akbd.h"
#include "bbc.h"
#include <stdarg.h>
```

```
void dprintf(int line,char * format, ...) {
  va_list args;
  char v[128];

  va_start(args, format);
  vsprintf(v, format, args);
  bbc_vdu(4);bbc_vdu(30);
  while(line--) bbc_vdu(10);
  printf("%-40s",v);
  bbc_vdu(5);
  if(akbd_pollsh()) bbc_get();
  va_end(args);
}
```

Please send us your C hints - all published hints will be paid for.