

NAME

a64l, l64a – convert between long and base-64

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <stdlib.h>
```

```
long a64l(const char *str64);
```

```
char *l64a(long value);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros(7)**):

a64l(), **l64a()**:

```
_XOPEN_SOURCE >= 500
```

```
|| /* glibc >= 2.19: */ _DEFAULT_SOURCE
```

```
|| /* glibc <= 2.19: */ _SVID_SOURCE
```

DESCRIPTION

These functions provide a conversion between 32-bit long integers and little-endian base-64 ASCII strings (of length zero to six). If the string used as argument for **a64l()** has length greater than six, only the first six bytes are used. If the type *long* has more than 32 bits, then **l64a()** uses only the low order 32 bits of *value*, and **a64l()** sign-extends its 32-bit result.

The 64 digits in the base-64 system are:

```
'.'    represents a 0
'/'    represents a 1
0-9    represent 2-11
A-Z    represent 12-37
a-z    represent 38-63
```

So $123 = 59 \cdot 64^0 + 1 \cdot 64^1 = "v/"$.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes(7)**.

Interface	Attribute	Value
l64a()	Thread safety	MT-Unsafe race:l64a
a64l()	Thread safety	MT-Safe

STANDARDS

POSIX.1-2008.

HISTORY

POSIX.1-2001.

NOTES

The value returned by **l64a()** may be a pointer to a static buffer, possibly overwritten by later calls.

The behavior of **l64a()** is undefined when *value* is negative. If *value* is zero, it returns an empty string.

These functions are broken before glibc 2.2.5 (puts most significant digit first).

This is not the encoding used by **uuencode(1)**.

SEE ALSO

uuencode(1), **strtoul(3)**