My view of glibc

Abstract：

1. Introduction

glibc is the libc library, or c runtime library, released by gnu, which is an integral part of gcc.The header files in glibc are the basis for all user-level application development.Therefore glibc is the main interface to OS control and the best interface besides system calls.

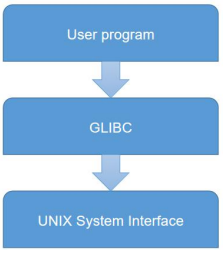
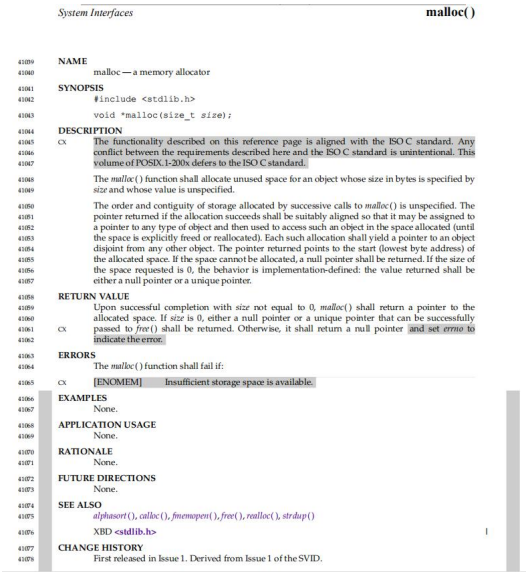


Figure1 the relation of GLIBC and system interface

History of GLIBC;

The glibc project was initially written mostly by Roland McGrath, working for the Free Software Foundation (FSF) in the 1980s. By 1992, it had the ANSI C-1989 and POSIX.1-1990 functions implemented and work was under way on POSIX.2. In September 1995 Ulrich Drepper made his first contribution to the glibc and by 1997 most commits were made by him. Drepper held the maintainership position for many years and until 2012 accumulated 63% of all commits to the project.

POSIX stands for Portable Operating System Interface. It's a family of standards specified by IEEE for maintaining compatibility among operating systems. Therefore, any software that conforms to POSIX standards should be compatible with other operating systems that adhere to the POSIX standards.The POSIX describes the functions, macros, and external variables to support applications portability at the C-language source level.It can be briefly said that the advent of posix gave birth to glibc, which led to glibc expanding on it to become even more powerful.



System Interfaces **malloc( )**

NAME

malloc — a memory allocator

SYNOPSIS

```
#include <stdlib.h>

void *malloc(size_t size);
```

DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-200x defers to the ISO C standard.

The *malloc*( ) function shall allocate unused space for an object whose size in bytes is specified by *size* and whose value is unspecified.

The order and contiguity of storage allocated by successive calls to *malloc*( ) is unspecified. The pointer returned if the allocation succeeds shall be suitably aligned so that it may be assigned to a pointer to any type of object and then used to access such an object in the space allocated (until the space is explicitly freed or reallocated). Each such allocation shall yield a pointer to an object disjoint from any other object. The pointer returned points to the start (lowest byte address) of the allocated space. If the space cannot be allocated, a null pointer shall be returned. If the size of the space requested is 0, the behavior is implementation-defined: the value returned shall be either a null pointer or a unique pointer.

RETURN VALUE

Upon successful completion with *size* not equal to 0, *malloc*( ) shall return a pointer to the allocated space. If *size* is 0, either a null pointer or a unique pointer that can be successfully CX passed to *free*( ) shall be returned. Otherwise, it shall return a null pointer and set *errno* to indicate the error.

ERRORS

The *malloc*( ) function shall fail if:

CX [ENOMEM] Insufficient storage space is available.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

alphasort( ), *calloc*( ), *fmemopen*( ), *free*( ), *realloc*( ), *strdup*( )

XBD **<stdlib.h>**

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Figure Posix defination

GLIBC is a part of GNU.GNU exists mainly because the early people with UNIX system used very well, UNIX system contains a variety of software, all can be used, but the good times are not long, unix for some projects to close the source and charge, resulting in users to open source design, so the name is called GNU Project. The GNU Project is a replacement for the kernel under UNIX and the tools and software on top of the kernel.Therefore the tools and software developed under GNU mostly follow unix rules and poxis rules, mainly because of the development in UNIX environment, which Leading to when Linux came along later, the interface was also designed according to unix. Therefore, the upper layer of glibc's interface is mainly according to poxic, the c language tool interface, and the lower layer is unix, the interface. The basic components of the system include the GNU compiler suite (GCC), the GNU C library (glibc), and the GNU core toolset (coreutils), in addition to the GNU debugger (GDB), and the GNU binary utilities (binutils).

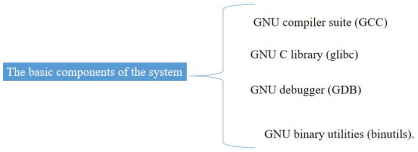


Figure 2 the basic components of the system

2. Content

(1) Virtual Memory Allocation And Paging;

(2) String and Array Utilities;

(3) Searching and Sorting;

(4) Pattern Matching;

(5) Input/Output on Streams;

(6) Low-Level Input/Output;

For historical reasons, the type of the C data structure that represents a stream is called FILE rather than “stream”.

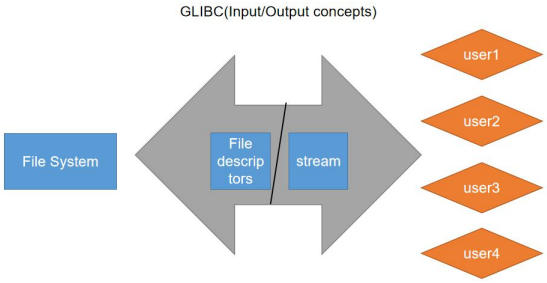


Figure 1.

3. Discuss

What principles is included in the design of a library;

(1) Header file for user calls;

The user call chain should be simple and straightforward.

(2) Not know the bottom-level interface;

Only let users care about the top-level interface and not the bottom-level interface.

(3) Functional design;

What a library should contain:

(1) User call;

User calls should explain to the user the basic implementation and functionality, as well as simple user logic. The description of the basic implementation method facilitates the user to select different interfaces to call.

(2) Internal implementation algorithm;

(3) Description of the dependent libraries;

Glibc

(4) Internal inter-call to facilitate the user's own coupling of most programs;

(5) Exposure of different levels of interfaces;

4. Conclusion

5. Acknowledgements