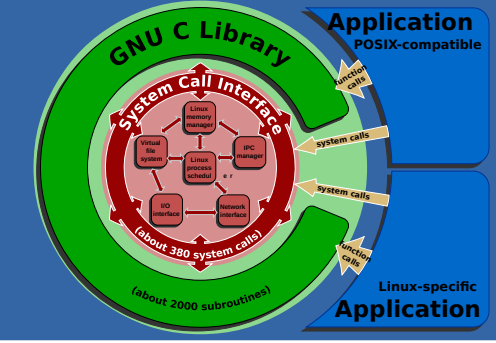
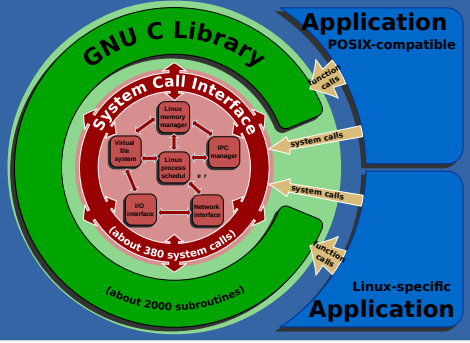


The year 2038 journey

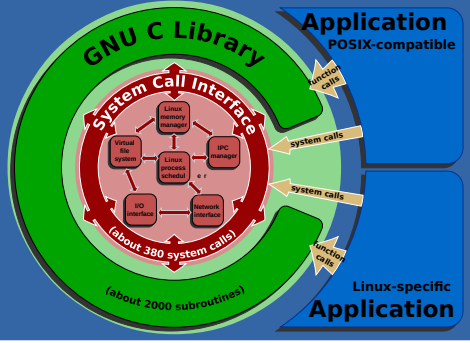
Agenda



By Shmuel Csaba Otto Traian, CC BY-SA 3.0,

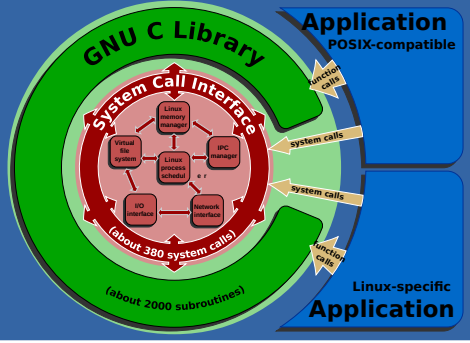
- Introduction
- Solution
- Development status
- For contributor
- Summary
- Discussion

Introduction – Łukasz Majewski



- Embedded SW engineer in DENX Software Engineering GmbH <lukma@denx.de>
- Electronics background
- Involved in projects:
  - U-Boot (DFU, USB)
  - Glibc (Y2038)
  - Zephyr (DSA)
  - Linux (Thermal, Power Management)
  - OE/Yocto (BSPs)

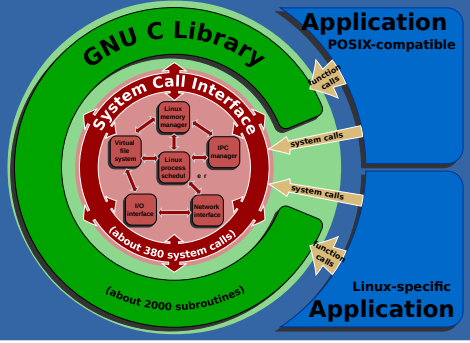
Introduction – Y2038 problem



By Shmuel Csaba Otto Traian, CC BY-SA 3.0,

- When?
  - Posix: 0x7FFFFFFF → 0x10000000
  - 19 Jan 2038 03:14:07 UTC → 13 Dec 1901 20:45:52
- Why?
  - The **time_t** variable (**signed int**) on 32 bit systems will overflow
- Affected
  - **32** bit **ARM, i386**, **PowerPC**, RISC-V
  - Heavy industry, railway, automotive, aerospace, medical, etc

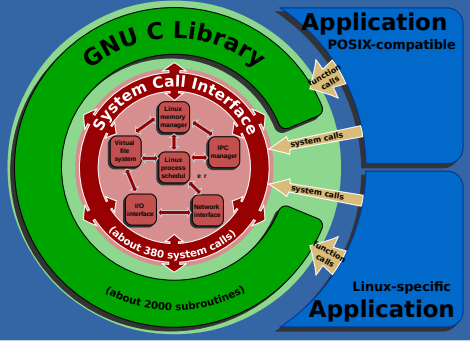
Introduction - glibc



By Shmuel Csaba Otto Traian, CC BY-SA 3.0,

- ~2 releases per year
- Conforms to
  - Unix 98
  - Single UNIX specification
  - POSIX (1c, 1d and 1j)
  - Partially ISO C99
- Ports
  - aarch64 **arm**
  - x86_64 (x32) **i386** ia64
  - alpha nios2 csky mips **PowerPC** RISC-V s390 sparc

Introduction – glibc eco-system

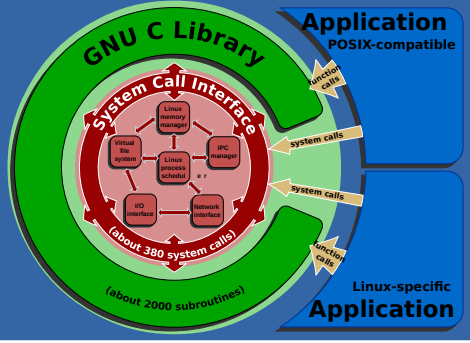


By Shmuel Csaba Otto Traian, CC BY-SA 3.0,

- Linux kernel binary - running on the system (it may or not support 64 bit time syscalls)
- Linux kernel headers, used at build time
  - `--enable-kernel` flag
  - The glibc knows which features are supported by kernel (proper flags are set based on version – e.g. `__ASSUME_TIME64_SYSCALLS`)
  - Proper exported system headers are provided
- Legacy user space programs
- Libraries installed on the system

Mixing above components on already deployed system may be catastrophic !!!

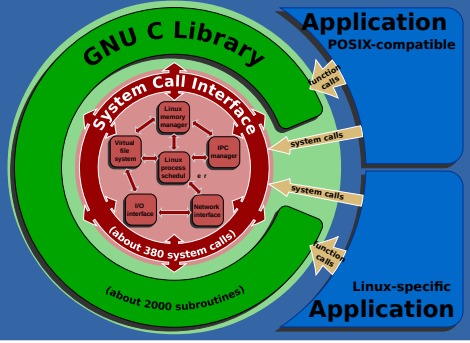
Solution (1)



By Shmuel Csaba Otto Traian, CC BY-SA 3.0,

- **Linux kernel**
  - In Linux v5.1 **new** time related syscalls, explicitly supporting 64 bit time, were introduced
    - For example for `clock_settime` → `clock_settime64` (it has new, unique syscall number – 112 vs. 404)
    - There are no functional changes for 64 bit machines (i.e. they still use `clock_settime`)
    - glibc would require, on the target system, kernel newer than **v5.1** to be Y2038 safe. Otherwise, it will fallback to syscalls supporting 32 bit time.

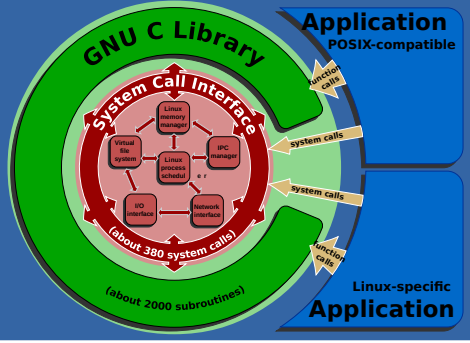
Solution (2)



By Shmuel Csaba Otto Traian, CC BY-SA 3.0,

- **ARM port design decisions**
  - No new port introduced for 64 bit `time_t`
  - Backward compatibility
  - Gradual conversion
  - New 32 bit CPU ports → minimal supported kernel must be v5.1
  - Add support for extra compilation flag `-D_TIME_BITS=64`
    - The LFS support (`-D_FILE_OFFSET_BITS=64`) is mandatory
  - Most likely scenarios to consider
    - Old Linux kernel (< v5.1) and contemporary glibc (v2.34)
    - New Linux kernel (> v5.1) and contemporary glibc (v2.34)
    - User programs compiled with `-D_TIME_BITS=64` running on glibc < v2.34

Solution (3)



By Shmuel Csaba Otto Traian, CC BY-SA 3.0,

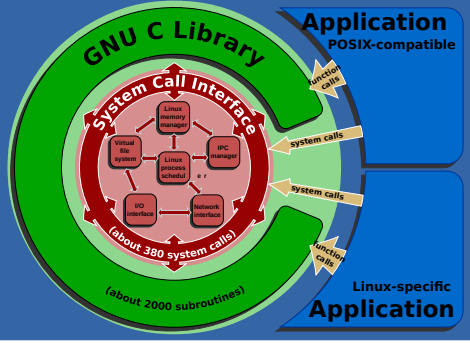
- **Glibc**
  - Convincing some key developers to support Y2038 work
  - All internal time related structures were refactored to support 64 bit time no matter on which port it runs (32 or 64 bit).

    On ARM, up till Y2038 rework, glibc was using 32 bit types (e.g. `struct timespec`)

    For example `struct timespec` → `struct __timespec64`
    - Prevents from ABI mismatches (alignment and structure size mismatch):

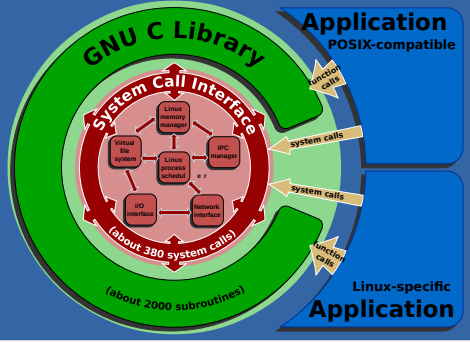
      user program → glibc → syscall
      (`__timespec64` → `timespec` → `__timespec64`)

Solution (4)



- **Glibc - __TIMESIZE macro**
  - It provides size of default `time_t` for target architecture
    - Default (ARM32 = 32, x86_64 = 64):
      `#define __TIMESIZE __WORDSIZE`
    - For new ports of 32 bit CPUs (arc, RISCV32):
      `#define __TIMESIZE 64`
      (64 bit time support from the very beginning of the port existence)
      - For ARM it cannot be set to 64 as it would break already deployed systems
      - Those legacy systems are defined in the internal code as:
        `__TIMESIZE != 64 && __WORDSIZE==32`

Solution (5)



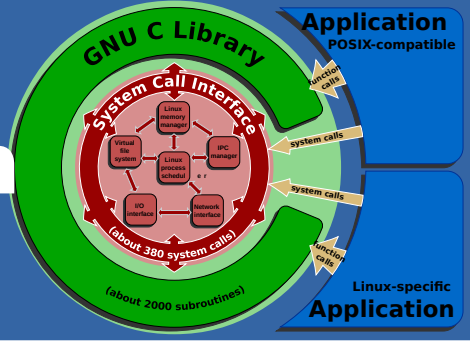
- **Glibc**
  - The `__USE_TIME_BITS64` flag from exported `/usr/include/bits/features-time64.h` enables redirection

Exportred in system's */usr*/include/sys/time.h

```
extern int gettimeofday (struct timeval *__restrict __tv,
                         void *__restrict __tz) __THROW __nonnull ((1));

#ifdef __USE_TIME_BITS64
# if defined(__REDIRECT_NTH)
extern int __REDIRECT_NTH (gettimeofday, (struct timeval *__restrict __tv,
                                          void *__restrict __tz),
                           __gettimeofday64);
# else
# define gettimeofday __gettimeofday64
# endif
#endif
```

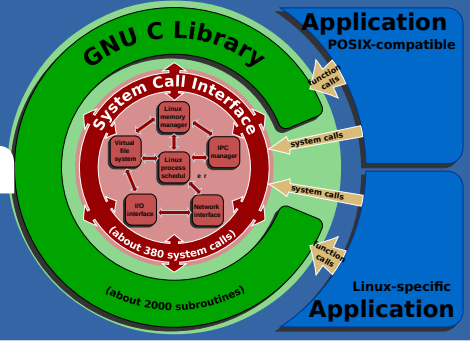
Solution (6) – syscalls conversion



Glibc's source *time/time.h*

```
#if __TIMESIZE == 64
# define __gettimeofday64 __gettimeofday
#else
extern int __gettimeofday64 (struct __timeval64 *restrict tv,
                             void *restrict tz);
libc_hidden_proto (__gettimeofday64)
#endif
```

Solution (7) – syscalls conversion



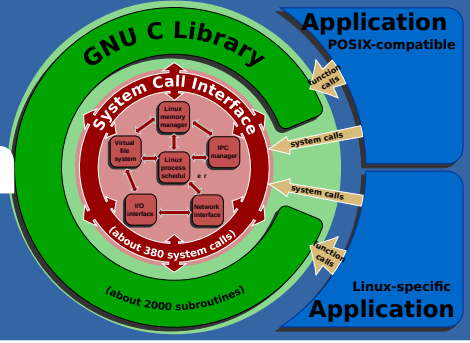
Glibc's source sysdeps/unix/sysv/linux/gettimeofday.c
(64/32 bit time support)

```
int
__gettimeofday64 (struct __timeval64 *restrict tv, void *restrict tz)
{
  if (__glibc_unlikely (tz != 0))
    memset (tz, 0, sizeof (struct timezone));

  struct __timespec64 ts64;
  if (__clock_gettime64 (CLOCK_REALTIME, &ts64))
        return -1;

  *tv = timespec64_to_timeval64 (ts64);
  return 0;
}
```

Solution (8) – syscalls conversion



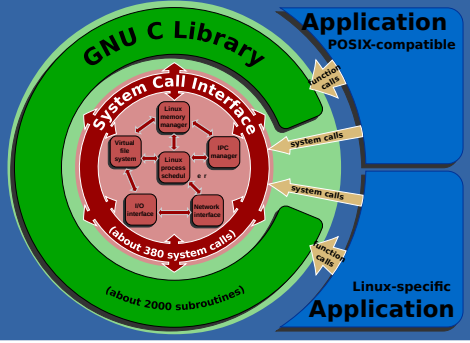

```
# if __TIMESIZE != 64
libc_hidden_def (__gettimeofday64)

int
__gettimeofday (struct timeval *restrict tv, void *restrict tz)
{
  struct __timeval64 tv64;
  if (__gettimeofday64 (&tv64, tz))
        return -1;

  if (! in_time_t_range (tv64.tv_sec))
    {
      __set_errno (EOVERFLOW);
      return -1;
    }

  *tv = valid_timeval64_to_timeval (tv64);
  return 0;
}
# endif
weak_alias (__gettimeofday, gettimeofday)
```

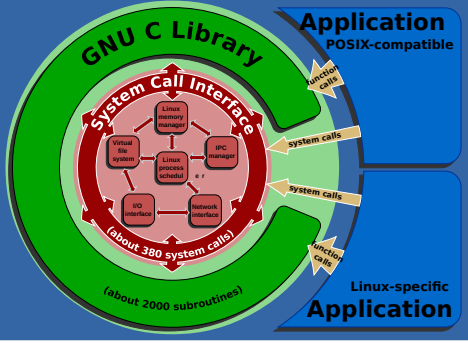
Y2038 development status (1)



By Shmuel Csaba Otto Traian, CC BY-SA 3.0,

- Code to make all relevant time related syscalls supporting 64 bit time has already been pulled.
- Only a few time related tests are not yet pulled (e.g. `tst-getrusage.c, tst-adjtime.c`)
  - All internal tests were ported to glibc test suite
- Glibc test suite now supports `--allow-time-setting` flag, which allows setting system time in a safe way
- There is the consensus that Y2038 support (`-D_TIME_BITS=64`) is going to be pulled for **2.34** release - though there will be some missing parts (like `utmp` conversion)

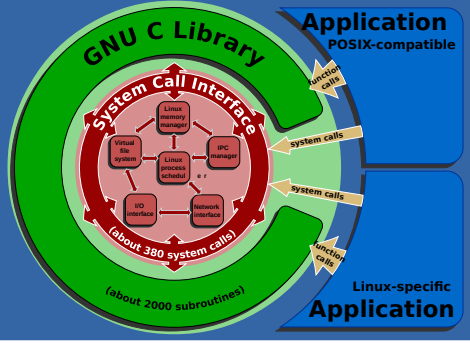
Y2038 development status (2)



By Shmuel Csaba Otto Traian, CC BY-SA 3.0,

- **Testing (Y2038):**
  - `build-many-glibcs.py`
    - Script to build `binutils`, `gcc`, `glibc` and run tests
  - Use QEMU with OE/Yocto
    - https://github.com/lmajewski/meta-y2038/
    - It is possible to change system date on the emulated ARM board (`--allow-time-setting`)
  - ABI compliance (`abi-compliance-checker`)

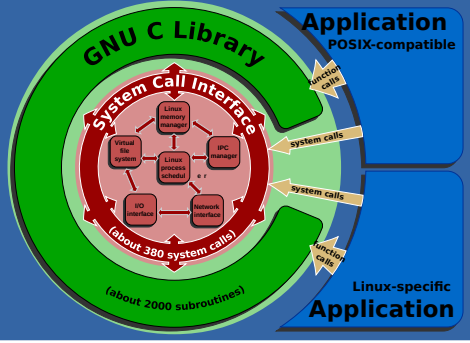
Y2038 development status (3)



By Shmuel Csaba Otto Traian, CC BY-SA 3.0,

- **Testing (Y2038):**
  - For Y2038 code acceptance all time related functions gained test coverage in `glibc` test suite (e.g. `tst-adjtime.c`)
  - Additionally, Y2038 specific tests (with `tests-time64` make target) were developed as well
    - Reuse the above code (`tst-adjtime-time64.c`) with
    - CFLAGS += -D_TIME_BITS=64 -D_FILE_OFFSET_BITS=64

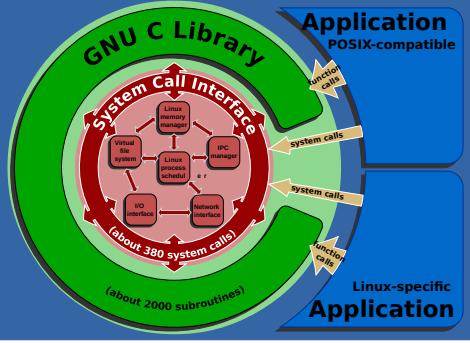
Y2038 development status (4)



By Shmuel Csaba Otto Traian, CC BY-SA 3.0,

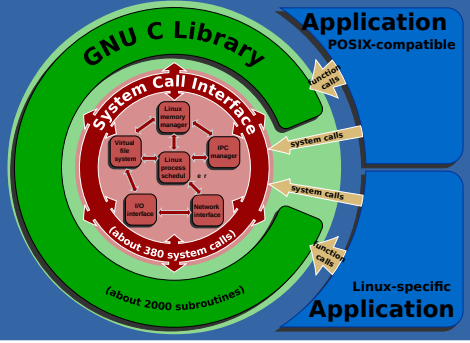
- **Future:**
  - Support for **64 bit time** on ports with **32 bit** CPUs **will be enabled by default** when glibc supported Linux is >= v5.1
    - Validation in legacy programs necessary
      - Check for `EOVERFLOW` errors
    - The code is now tested on ARM (QEMU)
      - Add support for i386 and PPC32
    - #Bugzilla: https://sourceware.org/bugzilla/enter_bug.cgi?product=glibc

For (Y2038) contributor (1)



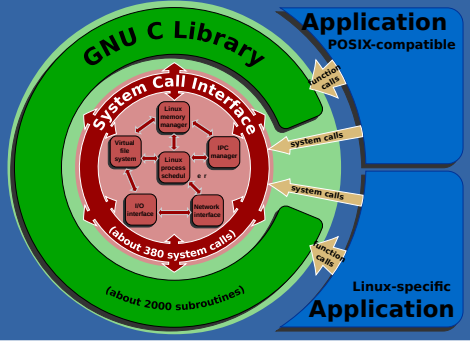
- **Paper work first**
  - It is not allowed to pull any code from a developer who did not sign "license agreement" contract with FSF

For (Y2038) contributor (2)



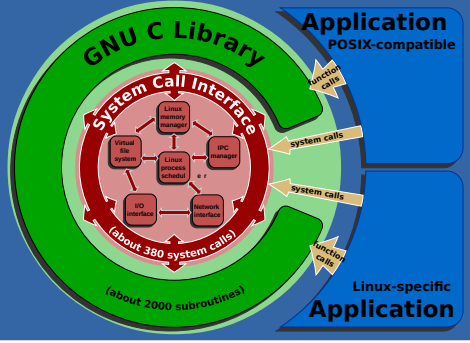
- **Key word: "consensus"**
  - Each developer before sending any patch is obliged to devise the "consensus" plan for the patch acceptance

For (Y2038) contributor (3)



- **Code upstreaming**
  - Each developer, after reaching the consensus in the community, is responsible for pushing the code to `-master` repository

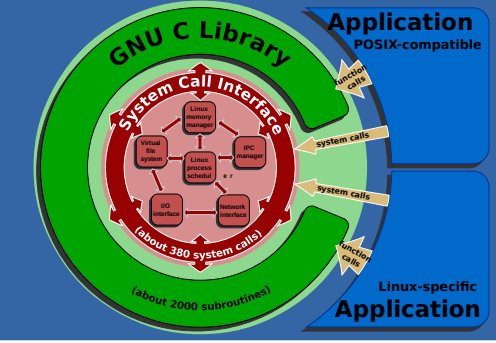
Summary



By Shmuel Csaba Otto Traian, CC BY-SA 3.0,

- The Y2038 problem is a **severe** threat
- Less than **17** years left for preparation
- Use **64 bit CPU** for new projects – no Y2038 issue
- If forced to use 32 bit CPU:
  - Use **-D_TIME_BITS=64** flag for compilation
  - Finally 32 bit time support in glibc (for 32 bit CPU) will be dropped

Questions/Discussions



By Shmuel Csaba Otto Traian, CC BY-SA 3.0,

- Questions/Discussion