# Operating Systems: Report 2

Joseph Jones (s2990652), Charles Randolph (s2897318),
Barnabas Busa (s2922673)

March 28, 2018

## Exercise 1: Modifying the Minix Kernel

---

#### Modifying the Startup Banner

The Minix startup banner/text was modified by augmenting the file `/usr/src/kernel/main.c` and added our names just after the copyright notice. This change is visible within the included `main.c` file. We applied these changes by recompiling with `make usage` and `make clean install`.

## Exercise 2: UTC time

---

#### Design of UTC Time

In order to properly implement a UTC time call, we must account for three possibilities regarding the system time. These may seem trivially obvious, but require a bit of consideration in the design of the program. In the following report, all mentions of "Epoch" refers to the date: `1/1/1970, 00:00:00 UTC`.

1. The system time is set to a date prior to Epoch.
2. The system time is set in between Epoch and the current date.
3. The system time is set in the future.

As mentioned in the assignment, it is not wise to subtract the current number of leap seconds from the system time alone. Doing so would not account for possibilities 1 and 2. Therefore, our algorithm stores a table of leap seconds since Epoch. Note that this table was lifted from the Wikipedia page concerning leap seconds, so the correctness of our computation will predicate on that. In order to calculate the number of leap seconds up to the current date, we subtract half-year increments of seconds from the current time in a loop. The loop continues this operation as long as the current year doesn't exceed the maximum table year (2018), and the remaining amount of seconds is non-negative. The former check avoids any out of bounds errors, and the latter avoids adding any leap seconds for dates prior to Epoch. It also ensures that leap seconds are added with half-year granularity.

Finally, we make sure to account for other phenomena, such as leap-years, when computing how many seconds to subtract over a year. We also make no assumptions about leap-seconds in the future.

### Implementation of UTC Time

UTC time was implemented as a system call (`do_utctime`) with an accompanying library wrapper (`utctime`). The library function was modelled off of the system `time` call, as it essentially serves the same purpose. Therefore, we placed the `utctime` header within `time.h` and borrowed `do_time`'s source code (from `time.c`) as a template for extracting the the system time in our system call. We simply applied our adjustment to the system time, and then attach it in the reply message to the library wrapper. The library wrapper finally returns this to the calling program as a `time_t` type. It also accepts a pointer to a `time_t` variable like `time` does, so that it may be optionally set there as well.

### Complete Installation Procedure

1. We first added `do_utctime` as entry number 69 in `/usr/src/servers/pm/table.c`.
2. We added the system call definition in `/usr/src/include/minix/callnr.h` as number 69 (`#define UTCTIME 69`).
3. We appended to the end of the `misc` section of `/usr/src/servers/pm/proto.h` our function prototype `int do_utctime(void); .`
4. We implemented our system call in `/usr/src/servers/pm`. The source file is titled `do_utctime.c`.
5. We modified the Makefile within `/usr/src/servers/pm` to include `do_utctime.c`.
6. To add a library function, we added a prototype (`time_t utctime(time_t *);`) to the library function `time.h` in `/usr/src/include/`.
7. In directory `/usr/src/lib/libc/sys-minix` we added the file `utctime.c` containing our library function. We made sure it imported it's header in `time.h`.
8. We then added `utctime.c` to the list of SRCS in `/usr/src/lib/libc/sys-minix/Makefile.inc`
9. We recompiled the kernel from `/usr/src` using `make build`
10. After the OS was rebuilt, we restarted and tested the new system call.

### Note

1. Installation of our changes can be made easy by simply running the included script `installAll.sh`. Please note that our shell script expects to find our files in `/mnt/shared/` within the Minix virtual-machine. We also supply a prebuilt `.ashrc` to make mounting easier. However, you will still need to setup the shared folder from the VirtualBox itself. Please refer to the appendix for doing so.
2. To test that our system call works, we have provided a sample program. If you run the installation script, it will be automatically placed in `/root`. Compile it with `cc -o utc_test /root/utc_test.c` and execute with `./utc_test` to see our system-call in action.

# Appendix

### Shared Folder Setup

Source

- Go to: Settings > Shared Folders > Create new folder share
- Add folder path
- Enter for example "shared" as the Folder Name
- Check Auto-mount
- Check Make permanent
- Click OK
- Start up the OS
- Edit the /root/.ashrc
- Insert the following code (if you use different folder name, then change the folder name everywhere)

```
[ -d /mnt/shared ] || mkdir /mnt/shared
mount | grep "/mnt/shared.*vbfs" > /dev/null ||
mount -t vbfs -o share=shared none /mnt/shared
```

- Now you can copy items to /mnt/shared folder and it is going to be visible on your local machine

**System call for utcTime**

```
#include "pm.h"
#include <minix/callnr.h>
#include <minix/com.h>
#include <signal.h>
#include "mproc.h"
#include "param.h"

/*
 ***************************************************************************
 *                                University of Groningen                  *
 * AUTHORS: Barnabas Busa, Charles Randolph, Joe Jones.                    *
 ***************************************************************************
*/

#define EPOCH           1970
#define SEC_PER_DAY     (24 * 3600)
#define DAYS_PER_YEAR   365
#define MAX_LEAPS       49

int leaps[MAX_LEAPS][2] = {
    {+0, +0},           // 1970
    {+0, +0},           // 1971
    {-1, -1},           // 1972
    {+0, -1},           // 1973
    {+0, -1},           // 1974
    {+0, -1},           // 1975
    {+0, -1},           // 1976
    {+0, -1},           // 1977
    {+0, -1},           // 1978
    {+0, -1},           // 1979
    {+0, +0},           // 1980
    {-1, +0},           // 1981
    {-1, +0},           // 1982
    {-1, +0},           // 1983
    {+0, +0},           // 1984
    {-1, +0},           // 1985
    {+0, +0},           // 1986
    {+0, -1},           // 1987
    {+0, +0},           // 1988
    {+0, -1},           // 1989
    {+0, -1},           // 1990
    {+0, +0},           // 1991
    {-1, +0},           // 1992
    {-1, +0},           // 1993
    {-1, +0},           // 1994
    {+0, -1},           // 1995
    {+0, +0},           // 1996
    {-1, +0},           // 1997
    {+0, -1},           // 1998
    {+0, +0},           // 1999
    {+0, +0},           // 2000
    {+0, +0},           // 2001
    {+0, +0},           // 2002
    {+0, +0},           // 2003
    {+0, +0},           // 2004
    {+0, -1},           // 2005
    {+0, +0},           // 2006
    {+0, +0},           // 2007
    {+0, -1},           // 2008
```

```
    {+0, +0},           // 2009
    {+0, +0},           // 2010
    {+0, +0},           // 2011
    {-1, +0},           // 2012
    {+0, +0},           // 2013
    {+0, +0},           // 2014
    {-1, +0},           // 2015
    {+0, -1},           // 2016
    {+0, +0},           // 2017
    {+0, +0},           // 2018
};

/* Returns one if the year is a leap year. Otherwise zero. */
static int isLeap (int y) {
    if ((y % 400) == 0) return 1;
    if ((y % 100) == 0) return 0;
    return ((y % 4) == 0);
}

/* Returns the number of seconds in a given year. */
static int secondsInYear (int y) {
    return SEC_PER_DAY * (DAYS_PER_YEAR + isLeap(y));
}

/* Returns the number of leap seconds since the epoch, for given elapsed seconds
    . */
static int leapSeconds (time_t elapsed) {
   int y = EPOCH, n = 0;
   while (y - EPOCH < (2 * MAX_LEAPS) && (elapsed - (secondsInYear(y) / 2)) > 0)
       {
       n += leaps[(y - EPOCH) / 2][y % 2];
       y++;
       elapsed -= (secondsInYear(y) / 2);
   }
   return n;
}

/* Returns the utc time given the current number of seconds since 1/1/1970 */
static time_t toUTC (time_t elapsed) {
    return elapsed + leapSeconds(elapsed);
}

/* Returns UNIX time adjusted for leap seconds */
int do_utctime(void) {
    clock_t uptime, boottime;
    int s;

    // Attempt to extract time.
    if ((s = getuptime2(&uptime, &boottime)) != OK) {
        panic("do_utctime couldn't get uptime: %d", s);
    }

    // Extract seconds since 1/1/1970.
    time_t unix_time = (time_t)(boottime + (uptime/system_hz));

    // Save adjusted time in message reply.
    mp->mp_reply.reply_time = toUTC(unix_time);

    return(OK);
}
```