Introducción a Sistemas Operativos: Ficheros

Clips xxx
Francisco J Ballesteros

1. El tiempo

UNIX mantiene su propia idea de la fecha y hora. Por un lado, el hardware habitualmente dispone de un reloj que está continuamente alimentado por baterías y que se ocupa de mantener la noción del tiempo.

Naturalmente, este reloj es un contador que se incrementa cada unidad de tiempo. Normalmente dispone de una serie de ticks por segundo y es programable respecto a la frecuencia a la que opera.

El sistema además suele programar el temporizador hardware para que cada `HZ` (una constante entera) veces por segundo genere una interrupción de reloj. Esto se usa, como ya mencionamos, entre otras cosas para expulsar del procesador aquellos procesos que llevan suficiente tiempo ejecutando (que han agotado su cuanto). Es tan simple como retornar de la interrupción en un proceso distinto.

Desde el shell, ya sabes que el comando *date(1)* informa respecto a la fecha y hora. Desde C, el principal interfaz para obtener el tiempo es *gettimeofday(2)* (y puede utilizarse *settimeofday(2)* para ajustarlo).

Esta función rellena una estructura de tipo *timeval* con los segundos y microsegundos desde una fecha convenida, llamada *epoch*. Normalmente desde el 1 de enero de 1970 en el caso de UNIX.

```
struct timeval {
        time_t       tv_sec;   /* seconds since Jan. 1, 1970 */
        suseconds_t  tv_usec;  /* and microseconds */
};
```

Además, rellena otra estructura llamada *timezone* que indica la zona horaria en que nos encontramos y si estamos en horario de verano (*daylight saving time*).

```
struct timezone {
        int     tz_minuteswest; /* of Greenwich */
        int     tz_dsttime;     /* type of dst correction to apply */
};
```

Pero *no deberías* dejar que `gettimeofday` rellene información sobre la zona horaria. Es mejor utilizar funciones de *ctime(3)* si deseas jugar con zonas horarias. Por ello nosotros vamos a utilizar `NULL` en el argumento correspondiente a la zona horaria para que `gettimeofday` lo ignore.

Este programa es similar a *date(1)*, pero con la opción "`-n`" imprime el número de segundos y microsegundos y los datos de la zona horaria en lugar de escribir la fecha normalmente.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <err.h>
#include <time.h>
#include <sys/time.h>

typedef unsigned long long uvlong_t;
static char *argv0;

static void
usage(void)
{
    fprintf(stderr, "usage: %s [-n]\n", argv0);
    exit(1);
}


int
main(int argc, char* argv[])
{
    struct timeval tv;
    int nflag;

    nflag = 0;
    argv0 = argv[0];
    if (argc == 2) {
        if (strcmp(argv[1], "-n") == 0) {
            nflag = 1;
        } else {
            usage();
        }
    }
    if (argc > 2) {
        usage();
    }

    if (gettimeofday(&tv, NULL) < 0) {
        err(1, "gettimeofday");
    }
    if (nflag) {
        printf("%llds %lldµs\n", (uvlong_t)tv.tv_sec, (uvlong_t)tv.tv_usec);
    } else {
        printf("%s", ctime(&tv.tv_sec));
    }
    exit(0);
}
```

La función *ctime(3)* se ocupa de generar un string para la fecha dada como un número de segundos desde *epoch*. Todas los argumentos de funciones de *ctime(3)* que aceptan un `time_t` suelen ser dicho número de segundos.

Para partir la fecha dada por un `time_t` (número de segundos desde *epoch*) y obtener el año, el mes, el día del mes, etc. normalmente se utilizan las funciones `localtime` (hora local) y `gmtime` (hora en

greenwich). Por ejemplo, como en

```
struct tm *t;
         struct timeval tv;
if (gettimeofday(&tv, NULL) < 0) {
    err(1, "gettimeofday");
}
t = localtime(&tv.tv_sec);
```

o bien

```
t = gmtime(&tv.tv_sec);
```

Y luego podemos usar los siguientes campos de la estructura `tm`:

```
int tm_sec;     /* seconds (0 - 60) */
int tm_min;     /* minutes (0 - 59) */
int tm_hour;    /* hours (0 - 23) */
int tm_mday;    /* day of month (1 - 31) */
int tm_mon;     /* month of year (0 - 11) */
int tm_year;    /* year - 1900 */
int tm_wday;    /* day of week (Sunday = 0) */
int tm_yday;    /* day of year (0 - 365) */
int tm_isdst;   /* is summer time in effect? */
```

Podríamos tener otros campos dependiendo del UNIX que usemos, pero seguramente no estén disponibles en todos los sistemas.

La función `mktime`, también documentada en *ctime(3)* hace el proceso inverso y genera un tiempo en segundos desde *epoch* a partir de una estructura de tipo `tm`.