POLITECHNIKA ŚWIĘTOKRZYSKA
Wydział Elektrotechniki, Automatyki i Informatyki
Katedra Urządzeń Elektrycznych i Automatyki

Systemy operacyjne czasu rzeczywistego

Zadania czasu rzeczywistego

Instrukcja laboratoryjna

Paweł Strączyński

2021

1 Tworzenie zadania RT w Xenomai (native API / alchemy API)

Zadanie jest tworzone poprzez wywołanie funkcji:

```
1        int rt_task_create(RT_TASK *task, const char *name, int stack_size,
2        int priority, int mode);
```

Gdzie odpowiednio:

1. *task* jest wskaźnikiem do struktury RT_TASK przechowującej informacje o zadaniu,
2. `name` to nazwa zadania,
3. `stack_size` to rozmiar stosu nowo utworzonego zadania czasu rzeczywistego,
4. `priority` jest priorytetem zadania (1-99),
5. `mode` jest to wektor flag wpływających na wykonanie zadania.

2 Uruchamianie zadania RT w Xenomai

W celu urchomienia zadania należy skorzystać z funkcji:

```
1        int rt_task_start(RT_TASK *task, void (*task_func) (void *), void *arg);
```

Gdzie:

1. `task` jest wskaźnikiem do struktury RT_TASK przechowującej informacje o zadaniu,
2. `task_func` jest to nazwa funkcji do wykonania przez zadanie czasu rzeczywistego,
3. `arg` to argument przekazywany do funkcji zadania.

3 Drukowanie RT do konsoli w Xenomai

W celu komunikacji z użytkownikiem programy RT mogą korzystać z biblioteki `rtdk`. Zawiera ona analogiczne funkcje do tych standardowych `printf()` jednak jest osadzona w przestrzeni użytkownika Xenomai i pozwala na drukowanie tak szybko jak to możliwe – bez używania jakichkolwiek blokad systemowych.

```
rt_vfprintf()
rt_vprintf()
rt_fprintf()
rt_printf()
```

Do inicjalizacji buforu `rt_print` służy funkcja

```
rt_print_auto_init(1);
```

4 Program przykładowy i Makefile

4.1 Program przykładowy 1

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <sys/mman.h>
#include <native/task.h>
#include <native/timer.h>
#include <trank/rtdk.h>

RT_TASK demo_task;

void demo(void *arg){
RT_TASK *curtask;
RT_TASK_INFO curtaskinfo;
rt_printf("Hello World!\n");

curtask=rt_task_self(); //odpytanie o biezace zadanie
rt_task_inquire(curtask,&curtaskinfo);
rt_printf("Task name : %s \n", curtaskinfo.name);
}

int main(int argc, char* argv[]){
        char str[10] ;
        rt_print_auto_init(1); //inicjowanie buforow rt_printf
        mlockall(MCL_CURRENT|MCL_FUTURE); //blokada swapowania pamieci
        rt_task_create(&demo_task, "hello", 0, 50, 0);
        rt_task_start(&demo_task, &demo, 0);
}
```

4.2 Program przykładowy 2

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <sys/mman.h>
#include <alchemy/task.h> //#include <native/task.h>
#include <alchemy/timer.h> //#include <native/timer.h>
#include <math.h>

#define CLOCK_RES 1e-9 //Clock resolution is 1 ns by default
#define LOOP_PERIOD 1e7 //Expressed in ticks

RT_TASK loop_task;

void loop_task_proc(void *arg)
{
  RT_TASK *curtask;
  RT_TASK_INFO curtaskinfo;
  int iret = 0;

  RTIME tstart, now;

  curtask = rt_task_self();
  rt_task_inquire(curtask, &curtaskinfo);
  int ctr = 0;

  //Print the info
  rt_printf("Starting task %s with period of 10 ms ....\n", curtaskinfo.name);

  //Make the task periodic with a specified loop period
  rt_task_set_periodic(NULL, TM_NOW, LOOP_PERIOD);

  tstart = rt_timer_read();

  //Start the task loop
  while(1){
    printf("Loop count: %d, Loop time: %.5f ms\n", ctr,
    (rt_timer_read() - tstart)/1000000.0);
    ctr++;
    rt_task_wait_period(NULL);
  }
}

```

```
43 int main(int argc, char **argv)
44 {
45   char str[20];
46
47   //Lock the memory to avoid memory swapping for this program
48   mlockall(MCL_CURRENT | MCL_FUTURE);
49
50   rt_printf("Starting␣cyclic␣task...\n");
51
52   //Create the real time task
53   sprintf(str, "cyclic_task");
54   rt_task_create(&loop_task, str, 0, 50, 0);
55
56   //Since task starts in suspended mode, start task
57   rt_task_start(&loop_task, &loop_task_proc, 0);
58
59   //Wait for Ctrl-C
60   pause();
61
62   return 0;
63 }
```

4.3 Przykładowy plik Makefile

```
1 XENO_CONFIG := /usr/xenomai/bin/xeno-config
2 CFLAGS := $(shell $(XENO_CONFIG) --native --cflags)
3 LDFLAGS := $(shell $(XENO_CONFIG) --native --ldflags)
4 CC := $(shell $(XENO_CONFIG) --cc)
5 EXECUTABLE := cyclic_test
6 all: $(EXECUTABLE)
7  %: %.c
8     $(CC) -o $@ $< $(CFLAGS) $(LDFLAGS)
```

5 Zadania do wykonania

1. Skompilować i uruchomić program przykładowy 1 z instrukcji laboratoryjnej.

2. Skompilować i uruchomić program przykładowy 2 z instrukcji laboratoryjnej oraz sprawdzić zasoby systemu w katalogu `/proc/xenomai`.

3. Rozbudować program z zadania drugiego o dwa dodatkowe zadania czasu rzeczywistego. Zbadać wpływ priorytetu zadania na jego wykonywanie.

4. Zmodyfikować programy z zadania 1 tak, aby korzystały z `POSIX API`