CSE4354/5354 Real-time Operating Systems
Fall 2024 Nano Project (Shell Interface)

1 Overview

The goal of this assignment is to refresh your skills in C while writing the shell interface to the operating system that will be used in the RTOS project.

2 Hardware Description

Microcontroller:
An ARM M4F core (TM4C123GH6PMI microcontroller) is required.

Serial interface:
If using the EK-TM4C123GXL evaluation board, then the UART0 tx/rx pair is routed to the ICDI that provides a virtual COM port through a USB endpoint.

3.3V supply:
The circuit is powered completely from the 3.3V regulator output on the evaluation board.

3 Software Description

A virtual COM power using a 115200 baud, 8N1 protocol with no hardware handshaking.

The shell interface should be designed to be scalable and extensible. It should handle commands with multiple arguments. Arguments can be numbers or characters. It should be case insensitive.

The code should not reference any string functions, such as sprintf, sscanf, or strtok, or strcmp, to minimize the code size and prevent potential problems with re-entrancy of C library functions (atoi and strlen are generally OK).

The shell must be written as an endless loop inside of a function called void shell(void). This code can call functions, except those in the C libraries. This will ensure that it will work when integrated with the rest of the operating system at a later time. A call to kbkit() before calling getcUart0(), with an empty function yield() being called in the pooling loop must be made when waiting for the next character to be received.

These are examples of the commands that will be supported in the final project:

Command	Function
reboot	Reboots the microcontroller. This will be implemented as part of the mini project
ps	Displays the process (thread) status. For now, it calls a function ps() which displays the text “PS called”
ipcs	Displays the inter-process (thread) communication status. For now, it calls a function ipcs() which displays the text “IPCS called”
kill *pid*	Kills the process (thread) with the matching PID. For now, it calls a function kill(uint32_t pid) which displays the text “*PID* killed” (PID is the number passed)
pkill *proc_name*	Kills the thread based on the process name
pi ON|OFF	Turns priority inheritance on or off. For now, it calls a function pi(bool on) that displays “pi on” or “pi off”.
preempt ON|OFF	Turns preemption on or off. For now, it calls a function preempt(bool on) that displays “preeempt on” or “preempt off”.
sched PRIO | RR	Selected priority or round-robin scheduling. For now, it calls a function sched(bool prio_on) that displays “sched prio” or “sched rr”.

pidof *proc_name*	Displays the PID of the process (thread). For now, it calls a function pidof(const char name[]) that displays “proc_name launched”.
proc_name	Runs the selected program in the background. For now, turn on the red LED.

4 Testing

Your code will be tested in the ERB 125 lab by the grader.

5 Deadlines

The assignment is due on the date and at the time indicated in the syllabus. This is an individual assignment. You may discuss the project with others, but sending or receiving code is not allowed.

6 Safety Issues

When utilizing the lab, please observe all safety rules as stated in the syllabus.

Have fun!