Μεταπτυχιακό Μάθημα ΟΣΥΛ

"Λειτουργικά Συστήματα Πραγματικού Χρόνου"

Μ.Στεφανιδάκης

Εργαστηριακή Άσκηση #3

Σκοπός της παρούσας άσκησης είναι η διερεύνηση του φαινομένου **αντιστροφής προτεραιοτήτων** και του πρωτοκόλλου **κληρονόμησης προτεραιότητας** κατά την αποκλειστική χρήση κοινών πόρων.

Εισαγωγή: Η εφαρμογή σας θα χρησιμοποιήσει τη βιβλιοθήκη LXRT του interface πραγματικού χρόνου RTAI στο λειτουργικό σύστημα Linux. Η βιβλιοθήκη αυτή επιτρέπει την ανάπτυξη εφαρμογών πραγματικού χρόνου σε **user space**. Για την δημιουργία της εφαρμογής σας θα συνδυάσετε τα εξής αρχεία:

α) βιβλιοθήκες RTAI: **rtai_hal rtai_lxrt rtai_sem rtai_msg rtai_mbx** (τα modules αυτά θα πρέπει να υπάρχουν στη μνήμη του συστήματος).

β) κώδικας εφαρμογής: **resource.c** (στο directory: **resource**).

1. Συνδεθείτε στο σύστημα του εργαστηρίου όπου έχει εγκατασταθεί το interface RTAI ως χρήστης **rtaiuser**. Δημιουργήστε ένα directory στο ίδιο επίπεδο με το resource.

2. Αντιγράψτε τα αρχεία από το directory **resource** στο δικό σας. Η εφαρμογή δημιουργεί 2 **περιοδικές** διεργασίες (T_1=14ms, C_1≈4ms, φ_1=3ms και T_2=14ms, C_2≈8ms, φ_1=0ms, priority1>priority2). Το αρχείο **resource.c** περιέχει τη συνάρτηση **main** του εκτελούμενου κώδικα, η οποία δημιουργεί τις διεργασίες πραγματικού χρόνου ως **παράλληλα threads,** καθώς και τον κώδικα για τις δύο διεργασίες (task1 και task2). Μεταγλωττίστε και εκτελέστε το πρόγραμμα. Σχεδιάστε το χρονικό διάγραμμα της εκτέλεσης από 0 έως 14 ms.

3. Τροποποιήστε τον κώδικα έτσι ώστε το task2 να χρησιμοποιεί έναν κοινό πόρο 2ms μετά την έναρξή του, και για διάστημα 4ms:

α) Δημιουργήστε μια semaphore για την αποκλειστική χρήση του κοινού πόρου στο κυρίως πρόγραμμα. Σε ποια τιμή πρέπει να αρχικοποιούνται πάντα semaphores που προστατεύουν κοινούς πόρους αποκλειστικής χρήσης; Φροντίστε επίσης να διαγράψετε τη semaphore στο τέλος του προγράμματος.

β) Τροποποιήστε το κώδικα της περιόδου του task2, έτσι ώστε να εκτελείται για 2ms (συνάρτηση dummy()), να δεσμεύει αποκλειστικά τον πόρο για 4ms (rt_sem_wait(locksem);), να τον απελευθερώνει (rt_sem_signal(locksem);) και να εκτελείται για ακόμα 2ms. Εκτελέστε το πρόγραμμα και σχεδιάστε το χρονικό διάγραμμα της εκτέλεσης από 0 έως 14 ms.

4. Τροποποιήστε τον κώδικα έτσι ώστε το task1 να χρησιμοποιεί τον κοινό πόρο 1ms μετά την έναρξή του, και για διάστημα 2ms, με τον ίδιον τρόπο όπως προηγουμένως. Σχεδιάστε το χρονικό διάγραμμα της εκτέλεσης από 0 έως 14 ms και υποδείξτε το χρονικό διάστημα κατά το οποίο παρατηρείται αντιστροφή προτεραιοτήτων.

5. Στο προηγούμενο ερώτημα η αντιστροφή προτεραιοτήτων είναι αναπόφευκτη, λόγω της αποκλειστικής χρήσης του κοινού πόρου. Στη συνέχεια δημιουργήστε μία ακόμα διεργασία task3 (T_3=14ms, C_3≈1ms, φ_3=5ms, priority1>priority3>priority2). Η διεργασία αυτή **δεν χρησιμοποιεί τον κοινό πόρο**. Σχεδιάστε το χρονικό διάγραμμα της εκτέλεσης από 0 έως 14 ms και υποδείξτε το χρονικό διάστημα κατά το οποίο παρατηρείται αντιστροφή προτεραιοτήτων. Τι συμβαίνει μεταξύ του task1 και task3;

6. Η αποφυγή αναμονής του task1 για το task3 αποφεύγεται μέσω του πρωτοκόλλου κληρονόμησης προτεραιότητας. Το RTAI υποστηρίζει το πρωτόκολλο αυτό, αλλά για να ενεργοποιηθεί πρέπει να χρησιμοποιήσετε μια semaphore ειδικού τύπου (resource type):

```
locksem = rt_typed_sem_init(nam2num("LCKSEM"),1,RES_SEM);
```

Σχεδιάστε πάλι το χρονικό διάγραμμα της εκτέλεσης από 0 έως 14 ms και βεβαιωθείτε ότι το task3 εκτελείται μόνο μετά το τέλος εκτέλεσης του task1.