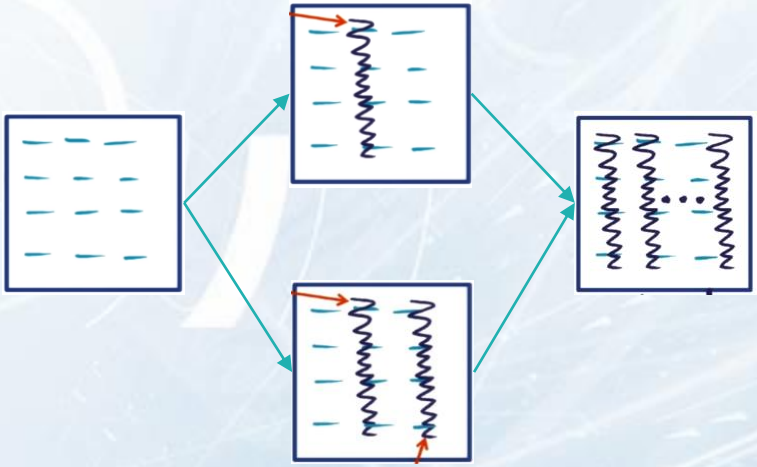
מערכות הפעלה תרגול 12

MUTEX

מתרגל-יורם סגל
yoramse@colman.ac.il

Program, Process and Thread

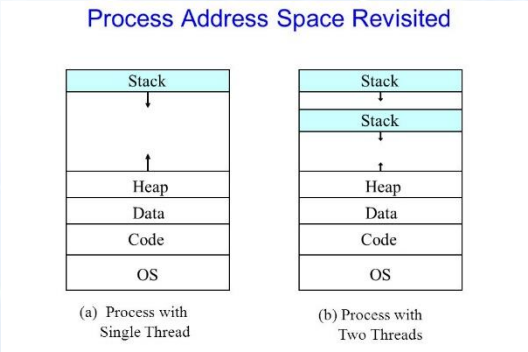


Process=Program + State of all threads executing in the program

הקדמה לחוטים

- חוט הוא יחידת ביצוע עצמאית בתוך תהליך.

- תהליך ב-Linux יכול לכלול מספר חוטים המשתפים ביניהם את כל משאבי התהליך :
  - מרחב הזיכרון.
  - גישה לקבצים והתקני חומרה.
  - מנגנונים שונים של מערכת ההפעלה.



- כל חוט בתהליך מהווה הקשר ביצוע נפרד – לכל חוט מחסנית ורגיסטרים משלו.

הקדמה לחוטים - יתרונות

- החוטים נועדו לאפשר ביצוע בלתי תלוי של חלקים מהמשימה של אותו תהליך.
- מספר חוטים של אותו תהליך יכולים לרוץ במקביל על **מעבדים שונים**.
- ניתן לשפר את ביצוע תהליך באמצעות שימוש בריבוי חוטים גם על מעבד יחיד

הקדמה לחוטים

- תהליך נוצר לראשונה עם חוט יחיד, החוט הראשי (primary thread).
- התקשורת בין חוטים של אותו תהליך היא פשוטה ביותר:
- הוספת חוט לביצוע תכנית זולה בהרבה מהוספת תהליך לאותה מטרה,

process descriptor

- לכל תהליך ב-Linux קיים בגרעין מתאר תהליך (process descriptor), שהוא רשומה המכילה:
  - מצב התהליך
  - עדיפות התהליך
  - מזהה התהליך (pid)
  - מצביע לטבלת איזורי הזיכרון של התהליך
  - מצביע לטבלת הקבצים הפתוחים של התהליך
  - ועוד..

הקדמה לחוטים

- התמיכה בחוטים ב-Linux שואפת להתאים לתקן הכללי של מימוש חוטים במערכות Unix הקרוי POSIX Threads (שבו כל החוטים של אותו תהליך מאוגדים ביחד).

- בלינוקס:
  - ההתייחסות לחוט הינה כאל תהליך רגיל ולכן לכל חוט אמור להיות **מתאר** משלו ו-**PID** משלו.
  - מצד שני, המתכנת, בהתאם לתקן POSIX, מצפה שלכל החוטים השייכים לאותו תהליך ניתן יהיה להתייחס דרך PID יחיד – של התהליך המכיל אותם.
    - ולכן הוא מצפה ש:
      - פעולות על ה-PID של התהליך ישפיעו על כל החוטים בתהליך.
      - פעולת ()getpid בכל חוט תחזיר את אותו ה PID (של התהליך המכיל את החוט).

קבוצת חוטים (thread group)

- הפתרון:

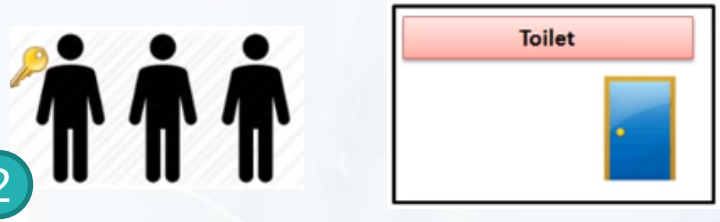
thread group

- כל החוטים השייכים לאותו תהליך נמצאים בקבוצה אחת.
- קבוצת החוטים מוגדרת בגרעין של Linux (מגרסת 2.4.X ומעלה) כדי לאפשר התייחסות משותפת לכל החוטים באותו תהליך.

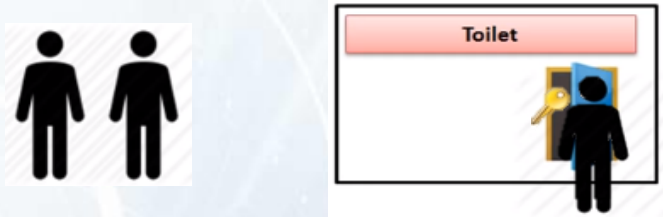
Semaphore Vs Mutex

Mutex

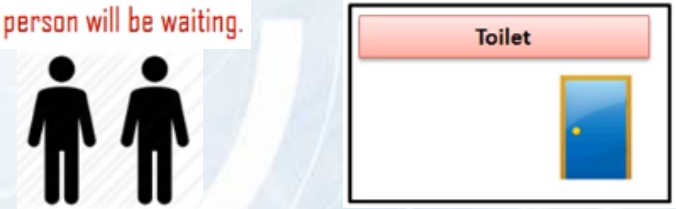
1 s assume toilet has a key and person who is having key can enter into the toilet.



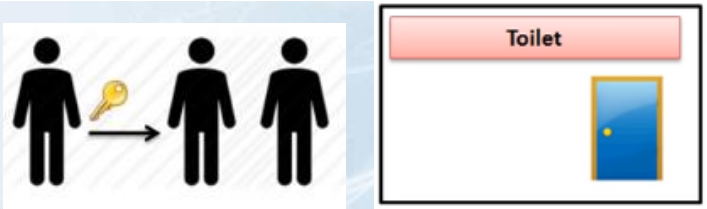
2 Person will unlock the door and then enter into toilet and and close it.



3 Other person will be waiting.



4 Now person will give it's key to next person and next person will enter into toilet.



Semaphore

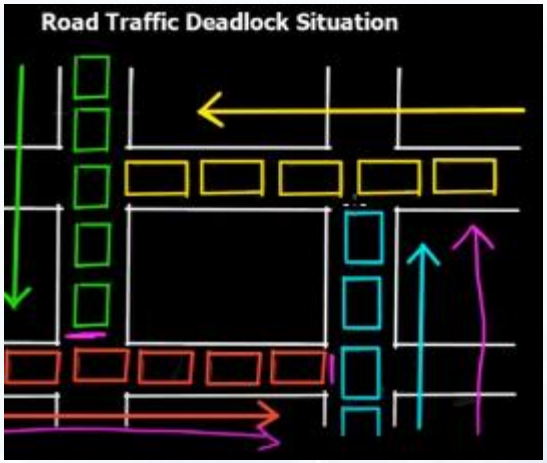
S=2

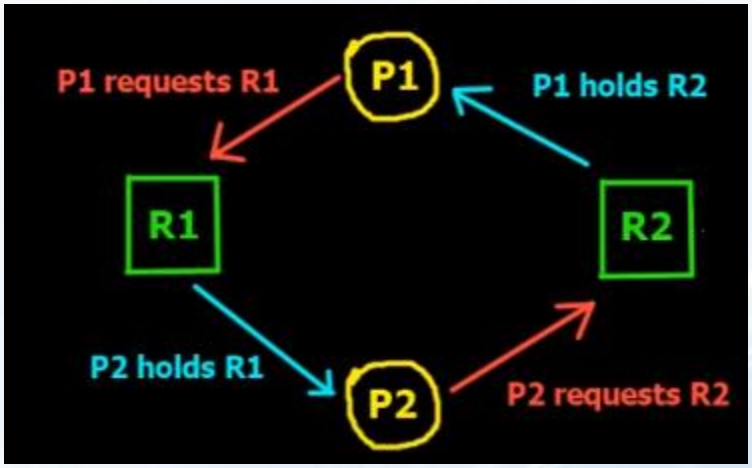
S=1

S=-1
Wait!!!

Yoram Segal

Deadlock





Methods for handling deadlock -

There are three ways to handle deadlock :

1) Deadlock prevention or avoidance -
   >> The idea is to not let the system into deadlock state.

2) Deadlock detection and recovery -
   >> Let deadlock occur, then do preemption to handle it once occurred.

3) Ignore the problem all together -
   >> If deadlock is very rare, then let it happen and reboot the system.
   >> Ignore the problem and pretend that deadlocks never occur in the system

3 Strategies to handle deadlocks :

1) Preemption -
   >> We can take a resource from one process and give it to other.
   >> This will resolve the deadlock situation, but sometimes it does causes problems.

2) Rollback -
   >> In situations where deadlock is a real possibility, the system can periodically make a record of the state of each process and when deadlock occurs, roll everything back to the last checkpoint, and restart, but allocating resources differently so that deadlock does not occur.

3) Kill one or more processes -
   >> This is the simplest way, but it works.

Mutex

❖מוטיבציה: t9_6.c

הדגמת בעיה שכמה חוטים מקדמים את המונה המשותף

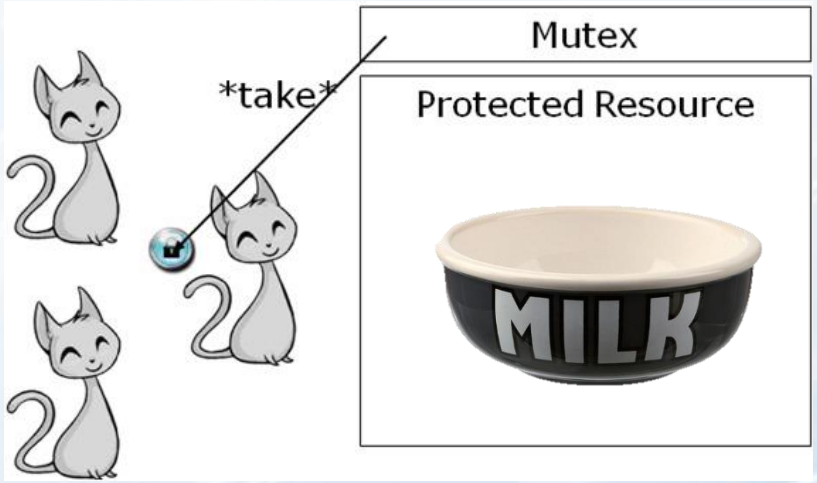
```
u2 89-231 69 : gcc -l pthread t10_6.c
u2 89-231 70 : a.out
counter value is: 3218187
u2 89-231 71 : a.out
counter value is: 3527220
u2 89-231 72 : a.out
counter value is: 6575929
u2 89-231 73 : a.out
counter value is: 6089293
u2 89-231 74 : a.out
counter value is: 4617099
u2 89-231 75 :
```

Mutex

- מנעול mutex מאפשר לחוט אחד בדיוק להחזיק בו (לנעול אותו).
  - כל חוט אחר המבקש להחזיק במנעול ייחסם עד אשר המנעול ישוחרר.
  - רק החוט המחזיק במנעול אמור לשחרר אותו (בעלות על המנעול).
- מנעולי mutex משמשים בדרך-כלל להגנה על גישה לנתונים משותפים, בתוך קטע קוד קריטי, ע"י נעילת המנעול בכניסה לקטע הקריטי ושחרורו בסופו.

Mutex


Mutex
Protected Resource
MILK


Mutex
take
Protected Resource
MILK


Mutex
Protected Resource
MILK


Mutex
Protected Resource
MILK
release

אתחול mutex

#include <pthread.h>

int pthread_mutex_init(pthread_mutex_t **mutex*, const pthread_mutexattr_t **attr*);

mutex- כתובת של אובייקט מסוג pthread_mutex_t

attr:

- **PTHREAD_MUTEX_NORMAL** - for ‘‘fast’’ mutexes
- **PTHREAD_MUTEX_RECURSIVE**
- **PTHREAD_MUTEX_ERRORCHECK**
- **PTHREAD_MUTEX_DEFAULT (or NULL)**
  - אתחול מנעול שכבר מאותחל – יגרור תופעה לא מוגדרת(תלוי ארכיטקטורת מעבד).
  - מומלץ לעבוד עם mutex מסוג "בודק שגיאות", כדי למנוע מצבים בעייתיים כגון אלו המסומנים באדום בשקף הבא

סוגי Mutex

סוג ה-mutex	נעילה חוזרת ע"י החוט המחזיק במנעול	שחרור מנעול ע"י חוט שאינו מחזיק במנעול	שחרור מנעול שכבר משוחרר
mutex מהיר	DEADLOCK	תוצאה לא מוגדרת	תוצאה לא מוגדרת
mutex רקורסיבי	הצלחה, מגדיל מונה נעילה עצמית ב-1	כשלון	כשלון
mutex בודק שגיאות	כשלון	כשלון	כשלון
mutex ברירת מחדל	לא מוגדר	לא מוגדר	לא מוגדר

הריסת mutex

- int pthread_mutex_destroy(pthread_mutex_t *mutex);

- הריסת מנעול גורמת לכך שלא יהיה אפשר להשתמש בו.
- כדי להשתמש שוב במנעול אפשר להפעיל עליו את הפונקציה pthread_mutex_init
- הריסת מנעול שנמצא במצב נעול או לא מאותחל תגרור תופעה לא מוגדרת.

נעילה, נסיון נעילה ושחרור

- נעילת mutex:

```
int pthread_mutex_lock(pthread_mutex_t *mutex);
```

  - הפעולה חוסמת עד שה-mutex מתפנה ואז נועלת אותו

- נסיון לנעילת mutex:

```
int pthread_mutex_trylock(pthread_mutex_t *mutex);
```

  - הפעולה נכשלת אם ה-mutex כבר נעול, אחרת נועלת אותו.

- שחרור mutex נעול:

```
int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

דוגמה: מנעולי mutex

```
pthread_mutex_t m;
int count;

void update_count() {
   pthread_mutex_lock(&m);
   count = count * 5 ;
   count++;
   pthread_mutex_unlock(&m);
}
```

```
int get_count() {
   int c;
   pthread_mutex_lock(&m);
   c = count;
   pthread_mutex_unlock(&m);
   return c;
}
```

1. מדוע צריך להגן על הגישה ל-count בתוך update_count()?
   כדי למנוע שיבוש ערך count בעדכונים מחוטים שונים.
2. מדוע צריך להגן על הגישה ל-count בתוך get_count()?
   כדי למנוע קבלת תוצאות חלקיות הנוצרות במהלך העדכון

שימו לב! גם אם ביטוי ההגדלה היה count++, לא מובטח שהקוד הנפרש באסמבלר הינו אטומי, ולכן יש להפעיל מנגנון סנכרון לפי הצורך.

נעילה, נסיון נעילה ושחרור

- t9_7.c

```
u2 89-231 82 : gcc -l pthread t10_7.c
u2 89-231 83 : a.out
counter value is: 10000000
u2 89-231 84 :
```

מקורות

- https://www.youtube.com/watch?v=O3EyzlZxx3g
- https://www.youtube.com/watch?v=DvF3AsTglUU
- https://www.youtube.com/watch?v=UVo9mGARkhQ