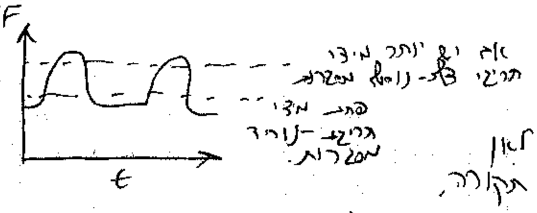
מערכות הפעלה - שיעור 7

* השיטה הכי מקובלת לפנות מקום היא בשני שלבים:
1. לבחור תהליך.
2. בתוכו מסגרת עצמו. זה נותן לנו יותר טוב זמנות מבחינת הפינוי.

* פינוי בשיטת LRU באופן גלובלי. פחות משתמשים רק ככה כיוון?

איך נבחר ~~מתהליך~~ מסגרת שאותו נרצה לפנות?

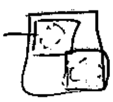


שיטה אמביוויוריסטית - הביאה לוניקס

בדרך כלל נסה לתת יותר מכונות. אבל מחשבים פה את השינוי.

בכל פעם נראה אם צריך מסגרות לקחת או להוסיף. למה שנוסיף? כי זה פחות תקורה, ופחות תקורה = יותר טוב.

יש פה בעיית הוגנות: מי שצריך יותר נותנים לו, צריך פחות לוקחים ממנו.
יש גם שיטה "סוציאליסטית" - לוקחים ממי שיש לו הכי הרבה.



הקבוצות שנלקחות. קריאה לפונקציה אולי בזמן לולאה.

צריך לחסום את הגודל של מה שצריך. נחשב את סה"כ השטח שצריך ונדע כמה מסגרות לתת.

התהליך הזה נקרא working set. זה הדפים שיש להם נטיה לעבוד בתוך הזמן מסוים.

2 3 2 4 2 2 3 7 0 4

צריך לקבוע חלון זמן

כל פעם שאנחנו מסתכלים כמה מסגרות צריך להסתכל כמה זמן אחורה אני מסתכל וכמה זמן אני אומר.

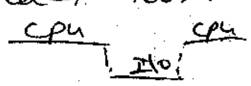
כבר לשים אסורים דרך המטרים על שיטה מהירה, יותר מקובל אחד.

כאשר אנחנו שקטע ממופה לזיכרון, פיזית צריך לעדכן את כל הדפים. וצריך גם לסמן בטבלה שזה invalid ולשים חלק מהדפים בזיכרון עם ה-invalid.

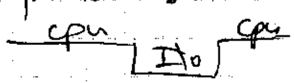
מפני משתמש ע"י מימוש הקוד משמש כל התהליכים. הקובץ פעם אחת בזיכרון וכל page table יש מיפוי משותף אפשר למנוע את הכתיבה המשותפת או לא מיפוי משותף מאפשר להעביר מידע בין תהליכים

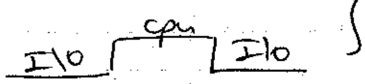
פרק 4 - תהליכים ותיזמון מעבדים

תהליך זקוק ל-cpu לפרק זמן מסוים. פרק הזמן הזה בד"כ נגמר (בתוכנית רגילה) עד שעושים IO (read, write). כשנגמר ה-IO שוב נרצה את ה-cpu.



אם היסטורית נשתמש ב-cpu כמשאב יקר, אז זמן ה-IO הוא זמן אבוד. האידיאל הוא שני תהליכים.





תיאור אידיאלי.

באופן מעשי אין מצב אידיאלי, אבל נריץ הרבה תהליכים במקביל ונוכל להשיג את "האידיאלי" הזה.

תהליך - מחשב וירטואלי, קולט את כל מה שנרצה ממחשב.
כאשר תהליך מושהה הוא באזור הדפדוף.

מתי מערכת ההפעלה משהה תהליך? כשהיא צריכה משהו, כמו IO

Read exception ← מערכת ההפעלה משהה התהליך ≡ רצה משהו אינסופי → יהיה לנו interrupt

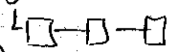
interrupt שעון - הוא תזמוני למערכת, יש לו קדימות על ה-cpu,
זו חומרה שעוצרת את הריצה.
מערכת ההפעלה הטיפוסית שמספיק השתמשו ב-cpu.

כאשר אני משהה תהליך: לא צריך לעשות כלום. הדפים בזכרון, ה-page table מצביע כמו שצריך. מה שכן, יכולים לסמן חלק מהדפים כנתונים לפינוי ולכתוב אותם על הדיסק אם הם "מלוכלכים".
ניהול סיגנלים, זה משהו שצריך להיות קיים.
נרצה לשמור גם את הנתונים הסטטיסטיים האחרונים של התהליך.

כשמערכת ההפעלה מחליטה להריץ תהליך היא תיצור לו מבנה נתונים שייצג את התהליך שרץ בזכרון.

למה צריך טבלה של משאבים זמניים?

Read (myfile ....)



הקישור עוברת
הרבה רמות במע' ההפעלה

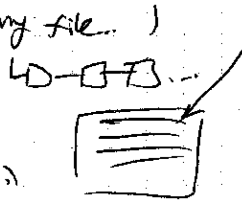
26, 325
31, 700

הרבה מבנים שמחזיקים איפה המשאב נמצא.

בד"כ לפני ה-read נבצע open(myfile) ואז תיבצע כניסה לטבלת המשאבים הזמניים. בהצלחה נקבל את זה בתור File Descriptor או handle.
הם מהווים אינדקסים למבנה הנתונים והמשאבים שיש לי.

השרשרת מקריאה ה-read עד למידע שאנו צריך להיות עבור המשתמש נרצה לנתק את הקשר מהמשתמש לאותו הניסיון עושה כבר שמקבל handle שמצביע לטבלה

read(my file...)



handle
(אינדקס לטבלת משאבים).

מערכת ההפעלה תבדוק אם נרצה לעשות שינוי לא נשנה.

אם הטבלה הזו היא טבלה per תהליך אז זה אפילו יותר מהיר.

מיתוג תהליכים - context switch

למה אי אפשר לעבור ישר מקוד של p1 לקוד של p2.
צריכים לשמור את המידע של p1. צריך לטעון את המידע של p2.
מישהו צריך לעשות את זה, זה ה-cpu.
איך אפשר לעשות זאת לבד? כי מערכת ההפעלה, אז נעשה זאת בחומרה.
כבר לא עוברת דרך מערכת ההפעלה. היו מחשבים שתמכו עם חומרה כזו.
למרות שזה נשבר בסוף של דבר (ההתקדמות שיצרה זאת של אז שהיו מחשבים מהירים.
מחשב של Intel, באילו מהירות מתבצעת מיתוג התהליכים?
זמן לגמרי מוגבלות. אי אפשר לעשות גישה לדיסק כי כל פעם יש מיתוג ואי אפשר שהתהליכים של זה תהיה כמו הזמן שלוקח לדיסק לרשום כי יש הרבה תקורה

תהליכים לעומת דפים - מבחינת התהליך הם של.

יש לתהליך: | cpu | mem | disk|...|
רוצו לפני כ-10 שנה שהרצתובר הזו לא מספיקה. כי הרבה פעמים זה קוד מורכב תהליכי. אם נותנים אותו שירות תמשיך אז אין נותן את אותו השירות להרבה - יש הרבה שכפול או זה בזבוז.

task1
task2
...

בעבר רק שנה איפה זה נמצא בזכרון. שכפול תהליך לאותה משימה זה בזבוז.
נרצה בתוך אותה מסגרת, ריבוי מעבדים וירטואליים. מרובה.
זה מה שמשיגים ע"י תהליכונים, חוטים. זה כמו מתשק מעבדים וירטואליים.
הם משתפים פרט למצב המעבד והנתונים של הזמון.
תהליכונים ב- threads זה כבר מוכן להיבנה מקבילית, גם אם יש לי מעבד 1. וגם יש לי הרבה מעבדים. זה כבר מעלה לא צריך להתחיל טבלת דפים - כי זה תואם התהליך המשותף עם יש חיסכון רב.
תהליכי (חוטי) ליבה - יש משימות של צרכי זמן משתנים. אם זה רץ על מבנה נתוני של מע' ההפעלה באיזה תהליך זה נשאר כרגע! גם צריך context... כי כל מי שלא רצה צריך זה הטבלה שלו page מונח הלוה 1 או 2

Kernel Threads