אוניברסיטת חיפה
החוג למדעי המחשב
מערכות הפעלה – תרגול

תרגול מס׳ 5

Memory Management (2)

1. שיטות להקצאת זיכרון

כאשר אנו מקצים זיכרון, לאחר זמן מה נוצרים לנו חורים בזיכרון (כתוצאה מזיכרון ששוחרר ועדיין לא הוקצה). בעיה זו נקראת Fragmentation.

1.1. שיטות לניהול הקצאת זיכרון

ישנן מס׳ שיטות לניהול הקצאת זיכרון.

1.1.1. Worst Fit

הזיכרון הבא יוקצה מהאזור הפנוי הגדול ביותר.

1.1.2. Best Fit

הזיכרון הבא יוקצה מתוך המקום המתאים ביותר להקצאה (מבחינת גודל).

1.1.3. First Fit

הזיכרון הבא יוקצה מתוך המקום הראשון המתאים להקצאה.

1.2. דוגמה

נבחן שלושה מקרים אשר כל אחד מראה את יתרונה של שיטת הקצאה מסוימת:
בכל דוגמא שטח הזיכרון הכללי האפשרי להקצאה הוא 1000 בתים ואנו מזניחים את הבתים שנדרשים לניהול הזיכרון.

1. התהליך מבצע פעמיים הקצאה של 100 בתים, שחרור של הבלוק הראשון שהוקצה, הקצאה של 100 בתים והקצאה של 800 בתים. Best fit **יצליח,** Worst fit לא יצליח, First fit לא יצליח (בהנחה שהבלוק ששוחרר נכנס בסוף רשימת האזורים הפנויים).
2. התהליך מבצע הקצאה של 100 בתים, 400 בתים, 200 בתים. כעת התהליך משחרר את הבלוק של ה-400 בתים, מקצה 100 בתים נוספים, משחרר את הבלוק של ה-200 בתים ומנסה להקצות 700 בתים. Best fit לא יצליח, Worst fit **יצליח,** First fit לא יצליח (בהנחה שהבלוק ששוחרר נכנס בסוף רשימת האזורים הפנויים).

3. התהליך מבצע הקצאה של 100 בתים, 200 בתים, 50 בתים, 100 בתים, 50 בתים ו-500 בתים. לאחר מכן משחרר התהליך את הבלוק של ה-200, הבלוק של ה-100 השני ובלוק ה-500 בתים. כעת, הקצאה של 100 בתים, שחרור בלוק ה-50 השני והקצאה של 650 בתים. Best Fit לא יצליח, Worst Fit לא יצליח, First Fit **יצליח** (אם הבלוק ששוחרר נכנס בסוף רשימת האזורים הפנויים).

2. זיכרון וירטואלי – המשך

להלן מס' מבני נתונים שתהליך משתמש בהם להקצאת זיכרון:

2.1. טבלת אזורים

לכל תהליך קיימת טבלה אשר מתארת את האזורים (Regions) השונים של התהליך. עבור כל אזור מכילה הטבלה את הנתונים הבאים:

1. **כתובת התחלת האזור** - כתובת וירטואלית בה מתחיל האזור.
2. **מספר הדפים באזור** - גודל האזור ביחידות של דפים.
3. **גישה לאזור** - האם ניתן לגשת לאזור ב- User Mode, האם ניתן לבצע Write באזור וכו'.
4. **הצבעה לטבלת דפים** - מצביע לטבלת הדפים של אזור זה.

2.2. טבלת דפים

לכל אזור וירטואלי, קיימת טבלת דפים אשר מכילה מיפוי של כתובת וירטואלית לכתובת פיזית. עבור כל דף מכילה הטבלה את הנתונים הבאים:

1. כתובת פיזית בה נמצא הדף.
2. אפשרויות גישה לדף - האם ניתן לגשת לדף ב-User Mode, האם ניתן לבצע Write לדף, האם הדף חייב להישאר בזיכרון ועוד.
3. דגלים לתמיכה במנגנון Demand Paging:

Valid - תוכן הדף חוקי (ניתן לשימוש).

Reference - התהליך פנה לאחרונה לדף זה.

Modify - תוכן הדף שונה ע"י התהליך.

Age - ה"זמן" שבו לא פנו לדף זה.

4. מיקום הדף על הדיסק - מיקום הדף על הדיסק יכול להיות בקובץ הדפדוף במקרה של דפי Data אשר שונו בזמן ריצתו של התהליך או בקובץ התוכנית במקרה של דפי Code.

=> טבלת הדפים + טבלת האזורים נקראות טבלאות תרגום.

קיימת גם רשימת דפים פיזיים פנויים - רשימה של דפים פיזיים (מסגרות) פנויים אשר אליהם ניתן לטעון דפים עפ"י בקשות מהתהליכים השונים.

2.3. אזורים משותפים

כפי שראינו, לכל תהליך טבלת אזורים כאשר כל אזור מצביע לטבלת דפים. כאשר רוצים לשתף נתונים בין תהליכים, מבצעים קישור ע״י הקצאת אזור בכל תהליך המצביע לטבלת הדפים אותה אנו רוצים לשתף. ז״א השיתוף נעשה ברמת האזור (המכיל מספר דפים וירטואליים). שיתוף זה שימושי לדוגמא עבור חלק ה-Text של תוכנית. מכיוון שלא ניתן לשנות חלק זה, כאשר מתבצע Fork, אין צורך להעתיק את טבלת הדפים של חלק זה אלא רק לציין שהאזור משותף וליצור בתהליך הבן אזור Text המצביע לטבלת הדפים אליה מצביע אזור ה-Text של תהליך האב (יפורט בהמשך).

2.4. פנייה לכתובת וירטואלית

כאשר תהליך פונה לכתובת וירטואלית, מתבצעות הפעולות הבאות:

1. החומרה ניגשת לטבלת האזורים של התהליך ומוצאת את האזור המתאים לכתובת. אם לא נמצא אזור מתאים או למשתמש אין זכות גישה, מתבצע Page Fault.
2. החומרה ניגשת לטבלת הדפים של האזור לכניסה המתאימה. אם הדף לא נמצא בזיכרון הפיזי או למשתמש אין זכות גישה, מתבצע Page Fault.
3. החומרה לוקחת את הכתובת הפיזית של תחילת הדף מתוך טבלת הדפים, מחשבת את הכתובת הפיזית הדרושה לפי החישוב:

   Physical Address = Page Start Address + Internal Offset

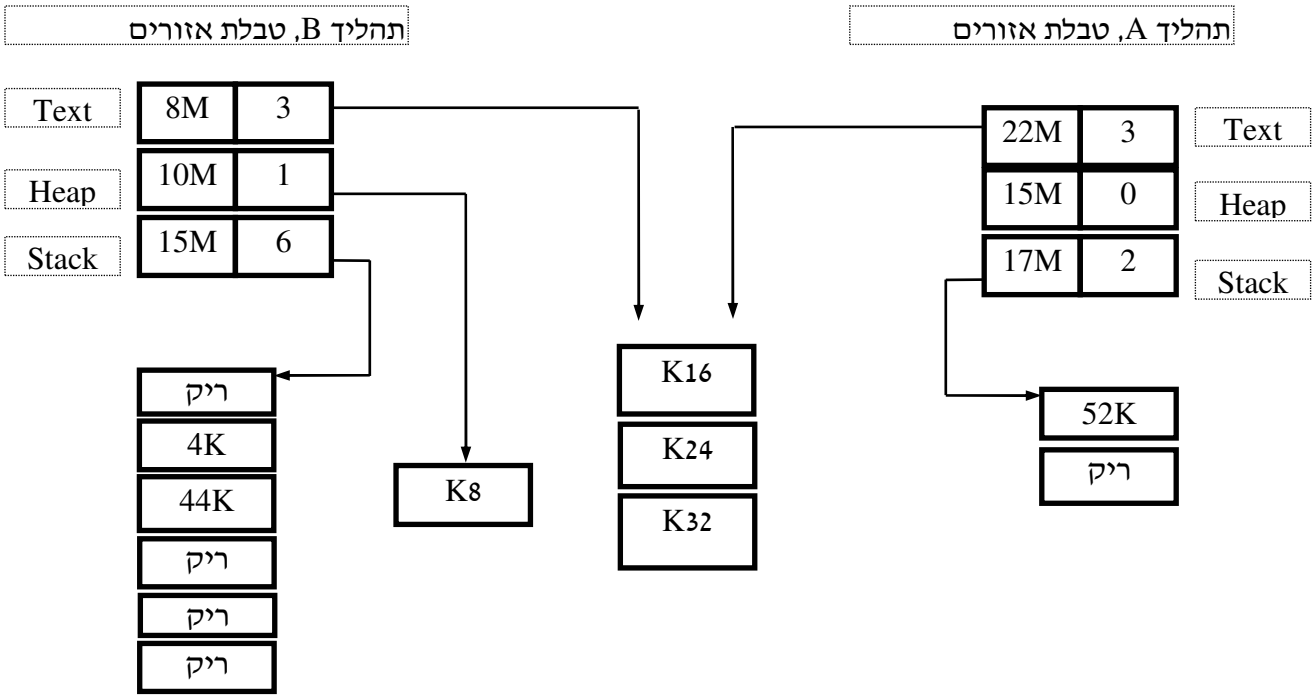
4. החומרה מדליקה את דגל ה-Reference בטבלת הדפים.
5. אם הפעולה היא Write, החומרה מדליקה את דגל ה-Modify בטבלת הדפים.
6. מתבצעת הפעולה המבוקשת.

במקרה של Page Fault מתבצעת קריאה למערכת ההפעלה כדי לטפל בבעיה שהתעוררה. במידה והבעיה היא דף שאינו בזיכרון הפיזי, עובר התהליך למצב sleep ונשלחת בקשה לתהליך מערכת לשם קריאת הדף המבוקש. אם הבעיה היא גישה לא חוקית, עוצר התהליך.
כאשר התהליך חוזר ממצב sleep מתבצעת מחדש הפקודה האחרונה (הפקודה שניגשה לזיכרון).

2.5. דוגמא

- כל דף בגודל 4K .
- אזור ה-Text משותף לשני התהליכים.
- כאשר תהליך A ניגש לכתובת וירטואלית 22M הוא יופנה לכתובת 16K בזיכרון הפיזי.
- כאשר תהליך B ניגש לכתובת וירטואלית 8M הוא יופנה לכתובת 16K בזיכרון הפיזי.
- כאשר תהליך B יפנה לכתובת 15M יקרה Page Fault אשר בעקבותיו יתבצע Swap In לדף המתאים לתוך הזיכרון.

- כאשר תהליך B יפנה לכתובת 15M+8K+20 הוא יופנה לכתובת הפיזית 44K+20.



2.6. תהליך ה-Pager

תהליך ה-Pager הוא תהליך מערכת הפעלה אשר מתעורר כאשר יש בקשות לטעינת דפים לזיכרון הפיזי.

כדי לטעון דף מבצע ה-Pager את האלגוריתם הבא:

1. מצא דף פיזי (מסגרת) פנוי מתוך רשימת הדפים הפיזיים הפנויים והקצה אותו.
2. טען דף מבוקש לזיכרון והדלק דגל Valid בטבלת הדפים של התהליך המתאים.
3. עדכן את הכתובת בטבלת הדפים של התהליך המתאים.
4. שחרר את התהליך הממתין לדף זה.

2.7. Page Stealer

תהליך מערכת הפעלה נוסף הוא ה-Page Stealer אשר מבצע Page Out לדפים אשר "הזדקנו" במערכת. התהליך מתעורר כל פרק זמן מסוים ומבצע מעבר על כל טבלאות הדפים של התהליכים השונים במערכת. התהליך מקדם את המונה Age עבור כל דף אשר מצוין כ-Valid בטבלת הדפים בתנאי שדגל Reference כבוי (לא הייתה פנייה לדף זה בזמן האחרון). אם דגל ה-Reference דלוק ה-Page Stealer מכבה אותו ומאפס את השדה Age. כאשר כמות הדפים הפיזיים הפנויים יורדת

מתחת לסף מסוים תהליך, זה מוסיף לרשימת הדפים הפנויים שנחשבים "זקנים" - מונה ה-Age שלהם גדול מסף מסוים. במידה ודגל ה-Modify של דף מסוים דלוק, הדף נכתב לדיסק.

3. ביצוע fork ו- exec במערכת Paging

נראה כעת כיצד מתבצעות קריאות ה-Fork וה-Exec במערכת מסוג Demand Paging:

3.1. Exec

בפעולת Exec, מערכת ההפעלה מאתחלת את טבלת האזורים וטבלאות הדפים כנדרש ומבצעת קפיצה לכתובת התחלת התוכנית. מכאן והלאה מתבצעת טעינה של דפים עפ"י דרישה. לדוגמא: ברור שכתובת התחלת התוכנית מצביעה לדף שאינו נמצא בזיכרון ולכן ייטען הדף לזיכרון במנגנון של Demand Paging שתואר.

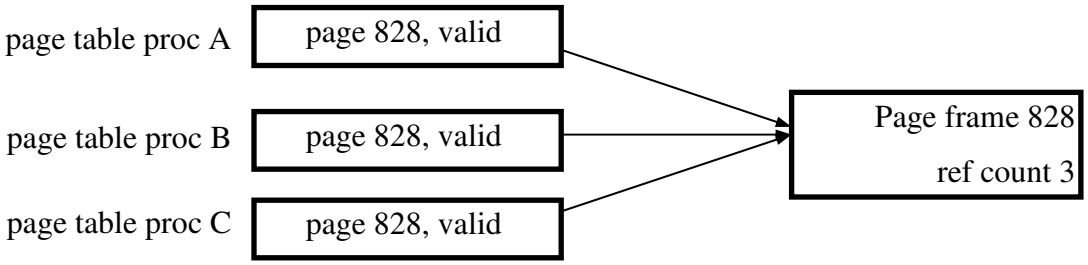
3.2. Fork

- בעקבות ביצוע קריאת המערכת fork, ה-kernel משכפל את טבלאות ה-Regions של תהליך האב עבור תהליך הבן.
  - עבור אזורים משותפים (למשל Text) רק מגדילים את מונה הפניות לאזור.
  - עבור אזורים פרטיים (כגון Data, Stack) מעתיקים את טבלאות הדפים של האב לטבלאות דפים חדשות (טבלאות הדפים של הבן).
- מעתה ניתן לפנות לדף דרך שני ה-Regions המשתפים אותו – האב והבן מצביעים לזיכרון פיזי זהה.
- כאשר אחד התהליכים מנסה לבצע כתיבה לדף מתבצע Page Fault. מערכת ההפעלה מזהה את גורם ה-Page Fault, יוצרת עותק חדש של דף זה עבור התהליך שמבצע את הכתיבה ומעדכנת את טבלאות הדפים שלו. לאחר הטיפול ב-Page Fault מתבצעת שוב פעולת הכתיבה בהצלחה.

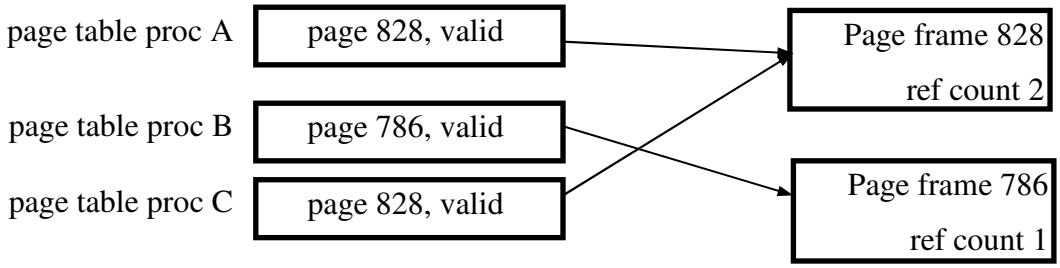
3.3. דוגמה

נניח ששלושה תהליכים משתפים דף פיזי מספר 828. תהליך B כותב לדף, וגורם ל-Fault-Protection בגלל שהגנת Read Only מופעלת. דף מספר 786 מוקצה, והתוכן של דף 828 מועתק אליו. מונה הפניות של דף 828 קטן ב-1. טבלת הדפים של תהליך B מעודכנת להצבעה על דף 786.

לפני ביצוע הכתיבה:



לאחר ביצוע הכתיבה:



4. אלגוריתמים להחלפת דפים בזיכרון פיזי

4.1. FIFO (First In First Out)

החלף את הדף שהוכנס ראשון לזיכרון.

למשל בזיכרון עם 3 דפים ועם סדר שימוש בדפים משמאל לימין:

7	**0**	**1**	**2**	**0**	**3**	**0**	**4**	**2**	**3**	**0**	**3**	**2**	**1**	**2**	**0**	**1**	**7**	**0**	**1**
7	7	7	2		2	2	4	4	4	0			0	0			7	7	7
	0	0	0		3	3	3	2	2	2			1	1			1	0	0
		1	1		1	0	0	0	3	3			3	2			2	2	1

4.2. Optimal

החלף את הדף שלא יהיה בשימוש הכי הרבה זמן.

7	**0**	**1**	**2**	**0**	**3**	**0**	**4**	**2**	**3**	**0**	**3**	**2**	**1**	**2**	**0**	**1**	**7**	**0**	**1**
7	7	7	2		2		2			2			2				7		
	0	0	0		0		4			0			0				0		
		1	1		3		3			3			1				1		

4.3. LRU (Least Recently Used)

החלף את הדף שלא השתמשו בו הכי הרבה זמן.

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		4	4	4	0			1		1		1		
	0	0	0		0		0	0	3	3			3		0		0		
		1	1		3		3	2	2	2			2		2		7		