1. אלגוריתם LRU בצורתו הטהורה לפינוי דפים דורש החזקת מבנה נתונים נוסף שתפקידו למיין את הדפים לפי כמו ההתייחסות אליהם, פעולה זאת של החזקת מבנה נתונים נוסף ושמירה עליו ממוין לאורך זמן דורשת תקורה גבוהה ולא משתלמם, ולכן זה לא יעיל להשתמש באלגוריתם LRU בצורתו הטהורה.
2. כן, דף יכול להיות בו זמניות בשני קבוצות עבודה, לדוגמא לאחר fork, קיימים 2 תהליכים שקבוצות העבודה שלהם שונות אבל מכילות דפים משותפים שהם כמובן גם זהים – עד שדף מסוים צריך להשתנות מאחד התהליכים ומפסיק להיות משותף לשניהם.

דוגמא נוספת היא שימוש בסיפריות שמשותפות לכמה תהליכים, במקרה זה הקוד של הסיפריה יהיה משותף לכולם.

ולכן זה לגיטימי במידה ויש כמה תהליכים שצריכים להשתמש בדף זהה, הוא יהיה משותף לכל הworkgoups שצריכים את הדף הזה ולא משוכפל לכל אחד בנפרד.

שימוש זה נקרא copy on write שדפים משותפים ביניהם לכמה תהליכים עד שאחד מהתהליכים משנה דף מסוים שמשתכפל ומופרד אוטומטית

1. לדוגמא במצב ש2 התהליכים ביחד צורכים יותר מכמות הזיכרון הראשית שאפשרית, ומה שקורה בפועל עם זיכרון וירטואלי הוא שבכל פעם שיהיה החלפה בין התהליכים ((context switch, הדפים שדרושים לתהליך שעובדים עליו יצטרכו לעבור מהזיכרון המשני לזיכרון הראשי ולהפך במקרה של התהליך השני, דבר זה ימנע אם לא היה נעשה שימוש בזיכרון וירטואלי, כל תהליך היה מבצע את העבודה שלו ומסיים אותה והיה מספיק פעם אחד להעביר דפים שקשורים לתהליך הבא, ולא כמה פעמים.

העברת דפים מהזיכרון הראשי למשני ולהפך היא פעולה יקרה שעולה הרבה זמן, ולכן מדובר בפגיעה די קשה במערכות שזמן/מהירות/ביצועים היא קריטית, כמו מערכות real time וembedded.

ולכן יש מערכות שאינן מפעילות כמה תוכניות בו זמין וסנכרון בין מספר תוכניות בזמן אמת לא קריטי להן, ולכן בוודאי שבמערכות כאלה יהיה עדיף להימנע משימוש בזיכרון וירטואלי.

1. הסקריפט:

וכך קיבלתי את הנתונים עבור המכונה שלי (אני לא בטוח שזו המכונה שסופקה על ידי האוניברסיטה, אבל גרסת האובונטו זהה)

כעת, נציב את הנתונים בנוסחה לחישוב הגודל האופטימלי של דפים (בהנחה שגודל שורה הוא 4 בתים)

כלומר גודל הדף האופטימלי במכונה הוא