

פרק 4

תהליכים ותזמון מעבדים

תהליכים

- ❖ תהליך הוא מחשב וירטואלי שמריץ תוכנית אחת עבור משתמש אחד
- ❖ תהליך יכול לרוץ על מעבד פיזי או להיות מושעה
- ❖ מערכת ההפעלה משעה תהליך כאשר הוא מחכה למשאב כלשהו (למשל לבלוק נתונים מקובץ או לטיפול בחריג דף) וכאשר צריך להריץ תהליך אחר במקומו

מבנה נתונים לייצוג תהליך

- ❖ מצב המעבד: ערכים של האוגרים, כולל מצביע התוכנית, כדי שאפשר יהיה להחזיר תהליך מושעה לריצה
- ❖ מפת זיכרון וירטואלי (טבלת דפים)
- ❖ טבלת שגרות איתות (signals; רק ביוניקס/לינוקס) שיופעלו כאשר מאותתים לתהליך או בזמן אירוע חריג. בחלונות יש מנגנון שונה לטיפול בחריגים
- ❖ זהות המשתמש המריץ והרשאות
- ❖ טבלת משאבים זמינים: קבצים פתוחים, קשרים פתוחים לתהליכים אחרים, וכדומה
- ❖ מצב תזמון וסטטיסטיקות

טבלת משאבים זמינים

- ❖ תהליך שמבקש משאב (גישה לקובץ, ערוץ תקשורת למחשב אחר, גישה להתקן חומרה) מקבל ממערכת ההפעלה מזהה משאב (handle בחלונות, file descriptor בלינוקס/יוניקס)
- ❖ המזהה הוא למעשה אינדקס בטבלת המשאבים של התהליך ששמורה בזיכרון שנגיש רק ממצב מיוחד
- ❖ כל איבר בטבלה של תהליך מכיל הרשאות (קריאה/כתיבה וכולי) ומצביע למבנה נתונים שמייצג את המשאב עצמו בזיכרון של מערכת ההפעלה
- ❖ ההצבעה הכפולה מונעת מתהליכים לזייף מזהה משאב
- ❖ חלק ממנגנון ההגנה של מערכת ההפעלה

מיתוג תהליכים

- ❖ כך מערכת ההפעלה מעבירה מעבד פיזי מתהליך אחד לאחר:
 - המעבד עובר מקוד של התוכנית לשגרה של מערכת ההפעלה כתוצאה מקריאת מערכת או פסיקה (כולל פסיקת שעון)
 - מערכת ההפעלה מטפלת באירוע, כלומר מבצעת את השירות שקריאת המערכת ביקשה או מטפלת בפסיקה
 - מערכת ההפעלה מחליטה שצריך להעביר את המעבד לתהליך אחר
 - מערכת ההפעלה שומרת את מצב המעבד במבנה הנתונים של התהליך כדי שניתן יהיה להמשיך את הרצתו
 - מערכת ההפעלה משחזרת את מצב המעבד של התהליך שצריך לרוץ
- ❖ זהו למעשה מנגנון מיפוי של מעבדים לתהליכים; אנלוגי למנגנון המיפוי של דפים למסגרות

תהליכים לעומת חוטים

(תהליכונים)

- ❖ חוט הוא מעבד וירטואלי
- ❖ תהליך מרובה חוטים מדמה מחשב וירטואלי מרובה מעבדים
- ❖ כל החוטים של תהליך משתפים את כל השדות במבנה הנתונים של התהליך פרט למצב המעבד וחלק ממצב התזמון
- ❖ בפרט, כל החוטים משתמשים באותה מפת זיכרון
- ❖ לכל חוט יש מחסנית משלו
- ❖ חוטים של תהליך עשויים לרוץ במקביל במחשב מרובה מעבדים או בזה אחר זה
- ❖ במיתוג בין חוטים של תהליך אין צורך להחליף את טבלת הדפים

תהליכי (חוטי) גרעין

- ❖ תהליכים של מערכת ההפעלה עצמה שניגשים רק למבני הנתונים שלה ולא לזיכרון של תהליך כלשהו
- ❖ מבני הנתונים של מערכת ההפעלה ממופים בצורה זהה לכל התהליכים ולכן תהליך גרעין יכול לרוץ עם כל טבלת דפים
- ❖ שדה טבלת הדפים במבנה הנתונים שלהם ריק
- ❖ בזמן מיתוג אליהם מערכת ההפעלה לא מחליפה טבלת דפים
- ❖ מיתוג כזה חוסך החלפה אחת או שתיים של טבלאות דפים
- ❖ תהליכי גרעין מועילים כאשר מעונינים שמטלות של מערכת ההפעלה יתחרו על משאבי מעבד עם תהליכים רגילים
- ❖ בלינוקס משמשים לפינוי מסגרות, למשל

יצירת תהליכים בלינוקס/יוניקס

- ❖ קריאת המערכת fork יוצרת תהליך חדש שהוא תאום זהה לתהליך הקורא פרט למספר התהליך; הקריאה חוזרת פעמיים
- ❖ לכל אחד מהתהליכים יש מרחב זיכרון זהה אבל פרטי
- ❖ מנגנון copy-on-write מאפשר לשכפל מרחב זיכרון בלי להעתיק את כל המסגרות:
 - הרשאת הכתיבה מוסרת מכל הדפים בשני התהליכים
 - ניסיון כתיבה גורם לחריג דף, מערכת ההפעלה מאבחנת את הסיבה, מעתיקה את המסגרת, ומשחזרת את הרשאת הכתיבה המקורית
- ❖ בחלונות תהליך חדש תמיד מריץ תוכנית מתחילתה

Zygote: שימוש מעניין ל-fork

- ❖ יישומים ב-Android ממומשים בג'אווה
- ❖ כל יישום רץ בתהליך נפרד
- ❖ התהליך הזה עושה הרבה עבודה לפני ששורת קוד אחת של היישום רצה:
 - אתחול המכונה הוירטואלית שמריצה תוכניות ג'אווה
 - טעינת כל הספריות הסטנדרטיות, ויש הרבה
 - אתחול כל אחת מהספריות, כולל יצירת מבני הנתונים שלהן
- ❖ Android יוצר תהליך אחד בשם Zygote שמאתחל מכונה וירטואלית עם כל הספריות וממתין בנקודה הזו
- ❖ יישומים מופעלים על ידי fork בתהליך הזה והרצת קוד היישום

מדיניות לתזמון מעבדים: מטרות

❖ שימוש יעיל במעבדים פיזיים: כדאי להמעיט במיתוג תהליכים

❖ זמן תגובה ראוי:

- לתוכנית אינטראקטיבית, זמן תגובה שייצור תחושה של תגובה מיידית בכפוף למגבלות החושיות של בני אדם
- לתוכניות ששולטות על מערכות פיזיות (מטוס, מפעל), זמן תגובה שמאפשר שליטה (לפני שהמטוס מתרסק)
- לסימולציית מזג אויר, סיום לפני מהדורת החדשות
- ...

❖ תוכניות שגרתיות (לא זמן אמת) אינן מכריזות על זמן תגובה נדרש לכל פעולה ולכן מערכת ההפעלה מסתפקת בתזמון הגון תוך התייחסות כללית לעדיפויות

מטרות תזמון משניות

- ❖ עמידה בלוחות זמנים: במערכות זמן אמת בלבד; מערכות הפעלה כלליות הן מורכבות מכדי להבטיח עמידה בלוחות זמנים (למשל בגלל חריגי דף)
- ❖ תזמון דביק: במחשב מרובה מעבדים, רצוי להריץ תהליך על המעבד שהריץ אותו לאחרונה, על מנת למרב את התועלת מה-TLB וזיכרונות המטמון; במחשב עם זיכרון מבוזר, כדאי להריץ תהליך על מעבד שקרוב למסגרות של התהליך
- ❖ תזמון כנופיות: כשמחשב מרובה מעבדים מריץ תוכניות מרובות חוטים, כדאי לתזמן הרצת כל החוטים של תהליך בבת אחת

מבני נתונים לתזמון מעבד:

multilevel feedback queues

max priority 0

--	--	--	--	--	--

 0.01 seconds max

			98	12	47
--	--	--	----	----	----

 0.01 seconds max

					11
--	--	--	--	--	----

 0.1 seconds max

min priority 0

				43	65
--	--	--	--	----	----

 10 seconds max

processes not ready to run

❖ התהליך בראש התור הגבוהה ביותר רץ

❖ מבנה הנתונים כולל כללים למעבר בין תורים

מעבר בין תורים

❖ מעבר בין תורים:

- תהליך שצבר זמן ריצה רב עובר לתור נמוך יותר
- תהליך שלא רץ זמן רב עובר לתור גבוה יותר, למניעת הרעבה
- תהליך שחיכה למשאב וקיבל אותו נכנס לתור גבוה מאוד כדי שישחרר את המשאב מהר ככל האפשר

❖ העדיפות שאנו מעניקים לתהליכים משפיעה על הפרמטרים של מעבר בין תורים ולא קובעת תור הספציפי לתהליך

❖ קל לממש מדיניות הוגנת ויעילה

❖ קל לממש יכולות מיוחדות, כגון תהליכים שרצים רק שאף תהליך אחר לא מוכן לריצה