מערכות הפעלה תרגול 9

Scheduling

מתרגל-יורם סגל
yoramse@colman.ac.il

תכולת התירגולים עד כה T1-T5

Q1

Pre-emptive & Non-Pre-emptive scheduling
Dynamically changing priorities
FCFS, LIFO, Aging

Q2

Round Robin (RR)
Priority Scheduling
Shortest Job First (SJF)

Q3

Multilevel Queue
אימיגרציה

תזכורת Non preemptive preemptive vs

- **Preemptive priority scheduling**
  - אלגוריתם המאפשר להפסיק את ביצוע קריאת מערכת system call באמצע הפעולה ולתת לתהליך בעל עדיפות גבוהה לבצע את קריאת המערכת שלו. בסיום היצוע קריאת המערכת של התהליך בעל העדיפות הגבוהה, מערכת ההפעלה תמשיך את ביצוע קריאת המערכת של התהליך בעל העדיפות הנמוכה.
- **Non-preemptive priority scheduling**
  - תהליך שהתחיל לבצע קריאת מערכת system call, יבצע אותה עד הסוף ולא חשוב מה העדיפות שלו.

שאלה 1

נתון אלגוריתם תזמון העדיפויות הבא (pre-emptive priority scheduling), המבוסס על סדרי עדיפויות, המשתנים באופן דינמי (dynamically changing priorities).

- ערכי עדיפות גדולים, מציינים עדיפות גבוהה יותר (כמו בWINDOWS).
- כאשר תהליך ממתין למעבד (ב-ready queue, בתור מוכן, אך אינו פועל), עדיפותו משתנה בקצב **אלפא** (כל יחידת זמן העדיפות גדלה באלפא) - Aging.
- כאשר הוא רץ, העדיפות שלו משתנה בקצב **בטא** (שינוי דינמי).
- בהתחלה, כל התהליכים מקבלים עדיפות של 0 כאשר הם נכנסים לתור-ל ready queue.
- ניתן להגדיר את הפרמטרים אלפא ובטא לאלגוריתמי תזמון שונים:

1. מהו האלגוריתם כאשר beta > alpha > 0 ?
2. מהו האלגוריתם כאשר alpha < beta < 0 ?
3. האם יש בעיית רעב ב -1? ב -2? יש להסביר.

שאלה 1 – סעיף 1

1. מהו האלגוריתם כאשר beta > alpha > 0 ?

נשים לב אלפא ובטא חיוביים.

כלומר העדיפות תמיד גדלה בין שהתהליך רץ ובין שהתהליך מחכה.

דוגמא:

בטא = 2, אלפא = 1,

3 תהליכים P1, P2, P3 מגיעים בזה אחר זה,

כל אחד מהם נמשך 3 מחזורי זמן.

Time	1	2	3	4	5	6	7	8	9
P1	0	2	4						
P2		0	1	2	4	6			
P3			0	1	2	3	4	6	8

התקדמות בבטא מסומנת בצהוב

התקדמות באלפא מסומנת בכתום

שאלה 1 – סעיף 1

1. מהו האלגוריתם כאשר beta > alpha > 0 ?

מסקנה:

מה שיש לנו כאן הוא אלגוריתם: First Come First Served

FCFS

(FIFO)

הוכחה
אם תהליך רץ אזי העדיפות שלו היא הגבוהה ביותר.
בזמן שהוא רץ, קצב התקדמות, העדיפות שלו, הוא הגדול ביותר (בטא), מכל תהליך אחר.
אם שני תהליכים ממתינים, אז עדיפותם תיגדל באותו קצב, כך שמי שהגיע ראשון לתור ההמתנה, יקבל ערך התחלתי גבוה יותר, ויקבל ראשון זמן מעבד.

שאלה 1 – סעיף 1

1. מהו האלגוריתם כאשר beta > alpha > 0 ?

מסקנה: מה שיש לנו כאן הוא אלגוריתם: FCFS - First Come First Served

Time	1	2	3	4	5	6	7	8	9
P1	0	2	4						
P2		0	1	2	4	6			
P3			0	1	2	3	4	6	8

ART – Average Response Time

AWT – Average Waiting Time

ATA – Average Turn Around

ART = (0+2+4)/(No. of Process) = (0+2+4)/3 = 2

AWT = ART זמן המתנה וזמן תגובה זהים כי לא יורדים לO/I

$P1_{TA}=3$ (רץ 3 מחזורי שעון)

$P2_{TA}=5$ (המתין 2 רץ 3 סה"כ 5 מחזורי שעון)

$P3_{TA}=7$ (המתין 4 רץ 3 סה"כ 7 מחזורי שעון)

ATA = $(P1_{TA} + P2_{TA} + P3_{TA})$/(No. of Process

ATA = (3+5+7)/3 = 5

שאלה 1 – סעיף 3 עבור סעיף 1

1. מהו האלגוריתם כאשר beta > alpha > 0 ?

מסקנה: מה שיש לנו כאן הוא אלגוריתם: FCFS – First Come First Served כלומר FIFO

נתון תהליכים סופיים CPU Bounded - כלומר כל תהליך זקוק לזמן CPU חסום.
האם FIFO יכול לייצר מצב של הרעבה?
תשובה: לא
הרעבה: תרחיש שבו תהליך לא יקבל לעולם CPU.
במקרה שלנו יכול לקחת הרבה זמן, אך עדיין, בסופו של דבר הוא יקבל CPU.

טענה: בהינתן שהתהליך הארוך ביותר צורך t זמן אזי כל תהליך שיגיע למערכת יקבל CPU.
הוכחה מילולית:
אם תהליך M נכנס למערכת, יש לפניו M-1 תהליכים לכל היותר (יכול להיות שיש פחות כי אולי חלק כבר סיימו).
מכיוון שהתהליך הארוך ביותר חסום ע"י t אזי התהליך הM יחכה (M-1)t – חסם עליון. לאחר זמן זה, בטוח שתהליך M יקבל זמן CPU.

הערה: במידה ותהליך לא חסום (תקוע בלולאה אין סופית), אזי יכולה להיווצר הרעבה, כי מדובר בFIFO

שאלה 1 – סעיף 2

2. מהו האלגוריתם כאשר alpha < beta < 0 ?

נשים לב אלפא קטן מבטא ושניהם שליליים.

כלומר העדיפות תמיד **יורדת** בין שהתהליך רץ ובין שהתהליך מחכה.

תהליך שרץ העדיפות יורדת ותהליך שרץ העדיפות יורדת יותר מהר, ככול שמחכים יותר.

מצב זה הוא Last In First Out (LIFO)
האחרון שמגיע לוקח את הCPU.

שאלה 1 – סעיף 2

2. מהו האלגוריתם כאשר alpha < beta < 0 ?

LIFO

דוגמא:

בטא (ריצה) = 1-, אלפא (המתנה) = 2-,

3 תהליכים P1, P2, P3 מגיעים בזה אחר זה (מתחילים בעדיפות 0),

כל אחד מהם נמשך 3 מחזורי זמן.

Time	1	2	3	4	5	6	7	8	9
P1	0	-1	-3	-5	-7	-9	-11	-13	-14
P2		0	-1	-3	-5	-7	-8		
P3			0	-1	-2				

התקדמות בבטא (ריצה) מסומנת בצהוב

התקדמות באלפא (המתנה) מסומנת בכתום

שאלה 1 – סעיף 2 - חישובים

ART – Average Response Time
AWT – Average Waiting Time
ATA – Average Turn Around

2. מהו האלגוריתם כאשר alpha < beta < 0 ?

מסקנה: מה שיש לנו כאן הוא אלגוריתם: LIFO

Time	1	2	3	4	5	6	7	8	9
P1	0	-1	-3	-5	-7	-9	-11	-13	-14
P2		0	-1	-3	-5	-7	-8		
P3			0	-1	-2				

ART = (0+0+0)/3 = 0 (LIFO) אלה אם שני תהליכים הגיעו באותו מחזור שעון

$P1_{TA}$=9 (צהוב + כתום)

$P2_{TA}$=6 (צהוב + כתום)

$P3_{TA}$=3 (צהוב + כתום)

ATA = ($P1_{TA}$ + $P2_{TA}$ + $P3_{TA}$)/(3)

ATA = (9+6+3)/3 = 6

$P1_{WT}$=6 (ממתין 6 מחזורי שעון - כתום)

$P2_{WT}$=3 (ממתין 3 מחזורי שעון - כתום)

$P3_{WT}$=0 (ממתין 0 מחזורי שעון - כתום)

AWT = ($P1_{WT}$ + $P2_{WT}$ + $P3_{WT}$)/(No. of Process

AWT = (6+3+0)/3 = 3

שאלה 1 – סעיף 3 עבור סעיף 2

2. מהו האלגוריתם כאשר alpha < beta < 0 ?

מסקנה: מה שיש לנו כאן הוא אלגוריתם: LIFO

נתון תהליכים סופיים CPU Bounded - כלומר כל תהליך זקוק לזמן CPU חסום.
האם LIFO יכול לייצר מצב של הרעבה?
תשובה: כן
הרעבה: תרחיש שבו תהליך לא יקבל לעולם CPU.
אם כל הזמן יגיעו תהליכים לא נגיע למי שמחכה.

<u>הוכחה מילולית:</u>
אם כל מחזור שעון יגיע תהליך חדש שצורך לפחות מחזור שעון חדש, או בצורה כללית אם כל X מחזורי שעון מגיע תהליך שצורך X מחזורי שעון, לעולם לא נגיע לתהליך שממתין. למעשה כל התהליכים יהיו בהרעבה כי כל פעם יגיע תהליך חדש.

הערה: יכולים להגיע אין סוף תהליכים. מדובר במצב מאד חמור כי מערכת ההפעלה לא מתקדמת. בפועל, מערכת ההפעלה תגיע לקצה גבול הקיבולת, ויכולה להיתקע.
מערכת ההפעלה אומנם מתחזקת בכל רגע נתון כמות סופית של תהליכים אך יש מצב שייוצרו אין סוף תהליכים בלולאה אין סופית).

שאלה 1 – סעיף 2 במצב Non preemptive

2. מהו האלגוריתם כאשר alpha < beta < 0 ?

LIFO

Non preemptive

דוגמא:

בטא (ריצה) = 1-, אלפא (המתנה) = 2-,

3 תהליכים P1, P2, P3 מגיעים בזה אחר זה (מתחילים בעדיפות 0),

כל אחד מהם נמשך 3 מחזורי זמן.

Time	1	2	3	4	5	6	7	8	9
P1	0	-1	-2						
P2		0	-2	-4	-5	-7	-9	-10	-11
P3			0	-2	-3	-4			

התקדמות בבטא (ריצה) מסומנת בצהוב

התקדמות באלפא (המתנה) מסומנת בכתום

שאלה 2

אצווה (BATCH) מכילה חמישה JOB-ים A, B, C, D ו- E אשר מגיעים למערכת ההפעלה כמעט באותו הזמן (A הגיעה ראשונה, E אחרונה, אך כולן הגיעו באותו מחזור שעון). הם בעלי זמני ריצה משוער (ניבוי סטטיסטי של מ"ע) של 10, 6, 2, 4 ו -8 מחזורי שעון. סדרי העדיפויות שלהם (שנקבעו חיצונית) הם 1,2,5,3 ו -4 בהתאמה, כאשר 5 היא העדיפות הגבוהה ביותר.

עבור כל אחד מאלגוריתמי התזמון הבאים, יש לקבוע את זמן ההחלפה הממוצע של התהליך ATA.

יש להתעלם ממשך זמן החלפה עצמו.

כל הJOB-ים דורשים ערך זמן מעבד חסום - CPU Bounded.

1. Round robin כאשר כל תהליך מקבל מחזור אחד (pre-emptive).
2. Priority Scheduling (non-pre-emptive)
3. FCFS (לפי סדר 10, 6, 2, 4, 8) (non-pre-emptive)
4. קדימות לJOB הקצר ביותר (non-preemptive)

שאלה 2 - ריכוז נתונים בטבלה

נתוני השאלה מרוכזים לטבלה

Process	Running Time	Priority
A	10	3
B	6	5
C	2	2
D	4	1
E	8	4

תזכורת :זמן כולל **TurnAround (TA)** של תהליך - כמה זמן עובר מאז הגיע התהליך ועד לסיומו.

Round robin לא עובד עם עדיפויות. הוא מסדר את התהליכים במעגל ונותן לכל תהליך לרוץ יחידת זמן מסוימת (`time quantum`)

שאלה 2 – סעיף 1 - Round robin

Round robin לא עובד עם עדיפויות. הוא מסדר את התהליכים במעגל ונותן לכל תהליך לרוץ יחידת זמן מסוימת (time quantum=1)

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Process	A	B	C	D	E(1)	A	B	C	D	E(2)	A	B	D	E(3)	A
	מחזור 1					מחזור 2					מחזור 3				
Time	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Process	B	D	E(4)	A	B	E(5)	A	B	E(6)	A	E(7)	A	E(8)	A(9)	A(10)
	מחזור 4			מחזור 5			מחזור 6			מחזור 7		מחזור 8		9	10

PC_{TA}=8 (המתנה + ריצה)

PD_{TA}=17 (המתנה + ריצה)

PB_{TA}=23 (המתנה + ריצה)

PE_{TA}=28 (המתנה + ריצה)

PA_{TA}=30 (המתנה + ריצה)

ATA = (8+17+23+28+30)/5 = 21.2

Process	Running Time	Priority
A	10	3
B	6	5
C	2	2
D	4	1
E	8	4

הערות לגבי - Round robin

- ✓ בRound robin ככול שמשך יחידת זמן הריצה גדולה יותר, יש פחות Content Switch.
- ✓ החיסרון המשמעותי של Round robin הוא שיש הרבה Content Switch.
- ✓ היתרון הוא תגובה מהירה, פשטות והגינות.
- ✓ נשים לב שבRound robin אין שימוש בעדיפות.

שאלה 2 – סעיף 2 - Priority Scheduling

(non-preemptive)

Priority Scheduling – התהליך בעל העדיפות הגבוהה ביותר ירוץ על הCPU. העדיפות נקבעת בתהליך חיצוני ע"י מערכת ההפעלה.

PBTA=0+6=6 (המתנה + ריצה)

PETA=6+8=14 (המתנה + ריצה)

PATA=14+10=24 (המתנה + ריצה)

PCTA=24+2=26 (המתנה + ריצה)

PDTA=26+4=30 (המתנה + ריצה)

ATA = (6+14+24+26+30)/5 = 20

Process	Running Time	Priority
A	10	3
B	6	5
C	2	2
D	4	1
E	8	4

שאלה 2 – סעיף 3 - FCFS

(non-preemptive)

סדר ההגעה: של התהליכים הוא A ראשון וE אחרון ובהתאמה זמני הריצה המשוערכים שלהם: A רץ 10 מחזורי שעון,B רץ 6, C רץ 2,D רץ 4,ואילו E רץ 8 מחזורי שעון

Process	Running Time
A	10
B	6
C	2
D	4
E	8

חישוב לפי סדר הגעה:

$PA_{TA}=0+10=10$ (המתנה + ריצה)

$PB_{TA}=10+6=16$ (המתנה + ריצה)

$PC_{TA}=16+2=18$ (המתנה + ריצה)

$PD_{TA}=18+4=22$ (המתנה + ריצה)

$PE_{TA}=22+8=30$ (המתנה + ריצה)

ATA = (10+16+18+22+30)/5 = 19.2

שאלה 2 – סעיף 4 - Shortest Job First (SJF)

SJF (non-preemptive)

פה עובדים לפי התהליך הקצר ביותר. (אין משמעות לעדיפויות)

Process	Running Time
A	10
B	6
C	2
D	4
E	8

$PC_{TA}=0+2=2$ (המתנה + ריצה)

$PD_{TA}=2+4=6$ (המתנה + ריצה)

$PB_{TA}=6+6=12$ (המתנה + ריצה)

$PE_{TA}=12+8=20$ (המתנה + ריצה)

$PA_{TA}=20+10=30$ (המתנה + ריצה)

ATA = (2+6+12+20+30)/5 = 14

ניתוח SJF ביחס למערכות הפעלה רגילות

SJF (preemptive)

ניתן להוכיח באינדוקציה כי אלג' SJF (preemptive) מבטיח מינימום ATA.

- אלג' SJF הוא אלג' אונליין : אלגוריתם שתוך כדי ריצה מקבל עוד אינפורמציה.
- אלג' SJF הוא אלג' המבוסס על חישוב הסתברותי

חסרונות

- סכנת הרעבה.
- אלגוריתם אונליין ולכן קיים קושי במדידה בזמן אמת של הזמן שכל תהליך דורש.
- מבוסס אלג' הסתברותי ויכול להיות שיש טעות בשערוך הזמן.

שאלה 3 – בעיית Multilevel queue

ארבע JOB-ים A, B, C, D מגיעים בו-זמנית. שיערוך זמני ריצה שלהם הוא 6, 10, 8 ו - 4 מחזורי שעון בהתאמה. נתונים שני תורים Q1 וQ2 אלג' המעבר בין התורים פועל בצורת **Round Robin** .

- לJOB-ים B, C יש עדיפות 1 עם ההגעה (כלומר נכנסים תמיד לQ1),
- ל-A, D יש עדיפות 2 (כלומר נכנסים תמיד לQ2).

יש לשרטט את דיאגרמת התזמון ולחשב זמן ATA אם אלגוריתם תזמון מבוסס על:

א) תזמון תורים מסוג Multilevel queue scheduling עם שני תורים:

- תור מספר 1 משתמש ב-**SJF** עם החלפה של כל 2 מחזורי שעון
- תור מספר 2 משתמש ב-**Round Robin** עם מחזור שעון אחד להחלפה
- ***אימיגרציה בין תורים יכולה להתבצע אך ורק בסיום מחזור החלפת אלגוריתמים***

ב) כמו א' אלא שהתהליך מועבר לתור השני, אם הוא נשמר בתור הנוכחי יותר מ -10 מחזורי שעון.

הערה - *בגרסה זו של תזמון תורים רב-שכבתי - המתזמן מרבב בין התורים. כלומר, בוחר תחילה תהליך מתור-Q1, ואז תהליך מתור- Q2, וחוזר חלילה. בכיתה למדתם גרסה אחרת, בה האלג' בתחילה מרוקן את כל Q1 מתהליכים, ורק לאחר מכאן מרוקן תהליכים מ-Q2.*

שאלה 3 - ריכוז נתונים בטבלה

נתוני השאלה מרוכזים לטבלה

Process	Running Time	Q1	Q2
A	6		X
B	10	X	
C	8	X	
D	4		X

תזכורת :זמן כולל **TurnAround (TA)** של תהליך - כמה זמן עובר מאז הגיע התהליך ועד לסיומו.

Round robin לא עובד עם עדיפויות. הוא מסדר את התהליכים במעגל ונותן לכל תהליך לרוץ יחידת זמן מסוימת (time quantum)

שאלה 3 – סעיף א'

Round robin לא עובד עם עדיפויות. הוא מסדר את התהליכים במעגל ונותן לכל תהליך לרוץ יחידת זמן מסוימת (time quantum=1)

Process	Running Time	Q1	Q2
A	6		X
B	10	X	
C	8	X	
D	4		X

נשים לב לאופן רישום התזמון:

מאד חשוב להקפיד להתחיל ממה מתחיל התזמון מ0 או מ1.

- אם מתחילים מ0 TA זה זמן סיום פחות זמן התחלה + 1
- אם מתחילים מ1 אזי TA זה זמן סיום פחות זמן התחלה + 2.

מחזור החלפה של SJF

מחזור החלפה של RR

Q1(SJF)	C1	C2		C3	C4		C5	C6		C7	C8		B1	B2		B3	B4		B5	B6		B7	B8		B9	B10		
Q2(RR)			A1			D1			A2			D2			A3			D3			A4			D4			A5	A6
Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

נזכור: בכל מחזור החלפה של אחד האלג' (SJF או RR) מחליפים בין תורים (בין Q1 לבין Q2)

PC_{TA}=11 (המתנה + ריצה)

PD_{TA}=24 (המתנה + ריצה)

PB_{TA}=26 (המתנה + ריצה)

PA_{TA}=28 (המתנה + ריצה)

ATA = (11+24+26+28)/4 = 22.25

שאלה 3 – סעיף ב' Multilevel feedback

ב) כמו א' אלא שהתהליך מועבר לתור השני, אם הוא נשמר בתור הנוכחי יותר מ -10 מחזורי שעון.

Multilevel feedback - תהליכים יכולים לנדוד בין התורים

כלומר כל 10 מחזורי שעון מבצעים הצרחה בין התכולה של התורים Q1 וQ2.

Process	Running Time	Q1	Q2
A	6		X
B	10	X	
C	8	X	
D	4		X

מחזור החלפה של SJF

מחזור החלפה של RR

בזמן החלפה תורים, D צרך 1 שעון, A צרך 2 שעונים. כלומר לD נשאר 3, לA נשאר 4 לכן בSJF D קיבל קדימות

Q1(SJF)	C1	C2		C3	C4		C5	C6		C7	C8		D2	D3		D4	A3		A4	A5		B5	B6		B7	B8	B9	B10
Q2(RR)			A1			D1			A2			B1			B2			B3			B4			A6				
Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Switch											+										+							

מחזור החלפת תכני תורים

מחזור החלפת תכני תורים

שאלה 3 – סעיף ב' Multilevel feedback

ב) כמו א' אלא שהתהליך מועבר לתור השני, אם הוא נשמר בתור הנוכחי יותר מ -10 מחזורי שעון.

Multilevel feedback - תהליכים יכולים לנדוד בין התורים

כלומר כל 10 מחזורי שעון מבצעים הצרחה בין התכולה של התורים Q1 וQ2.

Process	Running Time	Q1	Q2
A	6		X
B	10	X	
C	8	X	
D	4		X

מחזור החלפה של SJF

מחזור החלפה של RR

Q1(SJF)	C1	C2		C3	C4		C5	C6		C7	C8		D2	D3			D4	A3		A4	A5		B5	B6		B7	B8	B9	B10
Q2(RR)			A1			D1			A2			B1			B2				B3			B4			A6				
Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Switch											+										+							

מחזור החלפת תכני תורים

מחזור החלפת תכני תורים

$PC_{TC}=11$ (המתנה + ריצה)

$PD_{TA}=16$ (המתנה + ריצה)

$PA_{TA}=24$ (המתנה + ריצה)

$PB_{TA}=28$ (המתנה + ריצה)

ATA = (11+16+24+28)/4 = 19.75