جزوه ی سیستم عامل(جبرانی)

استاد فاطمه نجفی

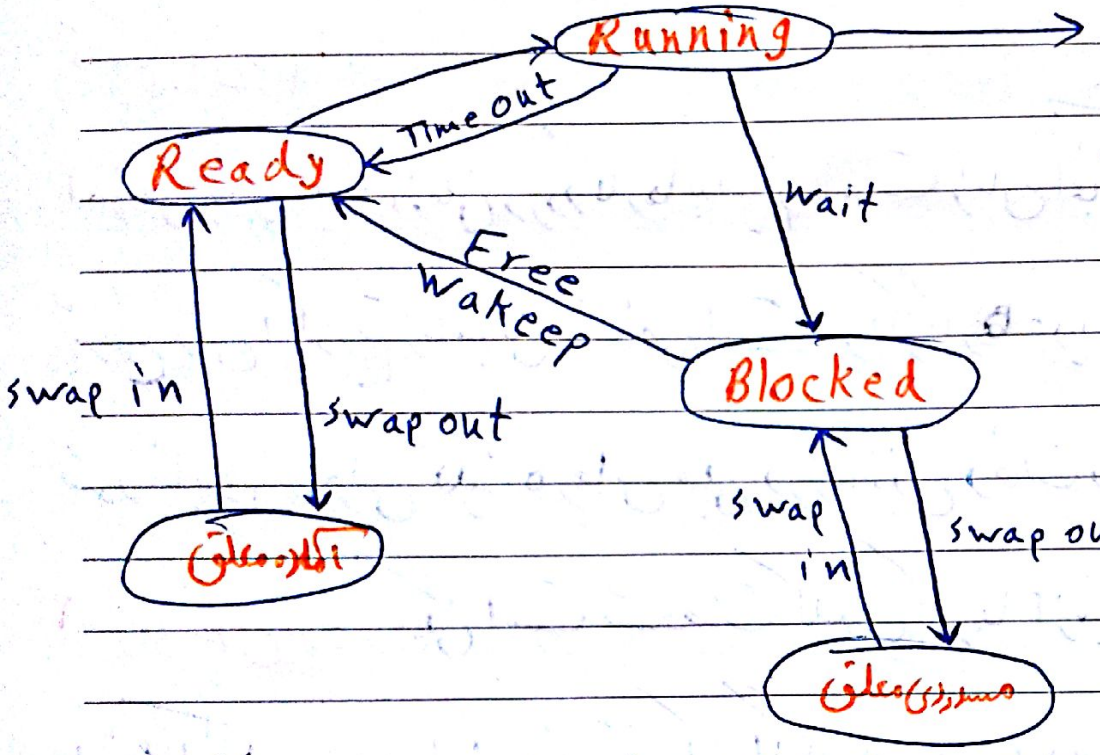
مدیر وبلاگ:

مهران مولایی

m-molaei.mihanblog.com

سیستم عامل (جبرانی)

Year:..... Month:..... Day:..... Subject:.....



زمانبندی: هدف از تخصیص پردازنده‌ها به فرایندها در طول زمان است به گونه‌ای که هدف‌های سیستم عامل برآورده شود.

انواع زمانبندی: ۱- زمان بندی انحصاری: هنگامی که پردازنده در اختیار یک فرایند قرار گرفت آن قدر در اختیارش باقی می‌ماند تا فرایند به اتمام برسد یا عمل I/O رخ دهد.

۲- زمان بندی غیر انحصاری: هنگامی که زمانبند سیستم عامل پردازنده را به یک فرایند واگذار کرده می‌تواند برطبق الگوریتم‌های زمان بندی برخلاف میل فرایند پردازنده را از فرایند پس بگیرد.

CACTUS

Scanned by CamScanner

Year:..... Month:..... Day:.....

Subject:.....

تقسیم بندی دوم:

۱- زمان بندی انحصاری: از زمان ورود کارها به سیستم و مدت زمان انجام آن ها قبل از شروع برای الگوریتم زمانبندی مشخص است و اگر کار جدیدی وارد سیستم شود تا پایان اجرای کلیهٔ کارهای قبلی در زمانبندی دخالت داده نمی شود.

۲- زمانبندی پویا: ممکن است مشخصات کلیدی کارها از قبل مدت زمان انجام آنها در زمان اجرای برنامه تغییر کند اگر کار جدیدی وارد سیستم شود در هنگام اجرای کارهای قبلی ممکن است در زمانبندی دخالت شود.

تقسیم بندی سوم:

۱- زمانبندی قطعی: هر بار که الگوریتم زمانبندی برای تعداد ورودی یا تعداد کار ثابت تکرار می شود یک جواب بدست می آید. مثال: HRN - RR - SJF

۲- زمانبندی غیر قطعی: در هر بار اجرای الگوریتم زمان بندی ممکن است جواب های متفاوتی بدست آید مانند الگوریتم ژنتیک.

CACTUS

Scanned by CamScanner

Year:..... Month:..... Day:..... Subject:.....

انواع سیستم های چند پردازنده ای:

از دیدگاه نرم افزاری به دو دسته متقارن و نامتقارن تقسیم می شود.

متقارن: در این سیستم ها تمامی پردازنده ها دارای نسخه یکسانی از سیستم عامل هستند و در هنگام نیاز با هم ارتباط برقرار می کنند.

نامتقارن: در این سیستم ها یا هر پردازنده سیستم عامل خاص خودش را دارد و یا اینکه یک سیستم عامل به بخش های شکسته شده و هر بخش روی یک پردازنده اجرا شود.

مزایای یک سیستم چند پردازنده ای: ۱- افزایش سرعت (به علت وجود اجرای موازی برنامه ها)

۲- افزایش قابلیت اطمینان (در صورت خراب یک پردازنده، پردازنده های دیگری توانند این کار را انجام دهند) ۳- کاهش نسبت هزینه به کارایی

مقایسه سیستم های چند کامپیوتر با چند پردازنده ای: ۱- هر دو سیستم به روش MIMD (چند داده - چند دستور) عمل می کند. ۲- ارتباط میان فرایندها در سیستم های چند پردازنده ای با استفاده از حافظه مشترک و در سیستم های چند کامپیوتری از طریق تبادل پیام صورت می گیرد

۳- سهولت قابلیت گسترش (در سیستم های چند کامپیوتری قابلیت گسترش بسیار راحت تر از

Scanned by CamScanner

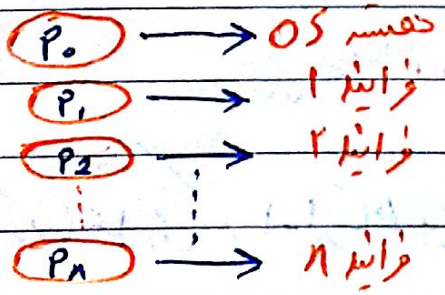
Year:..... Month:..... Day:..... Subject:.....

چندپردازنده‌ای است). ۴- روی سیستم‌های چندپردازنده‌ای یک نوع سیستم عامل وجود دارد اما در سیستم‌های چندکامپیوتری روی هر کامپیوتر سیستم عامل متفاوت وجود دارد.

۵- سرعت (سرعت در سیستم‌های چندپردازنده نسبت به چندکامپیوتری بیشتر است)

سیستم عامل‌های چندپردازنده‌ای: از جمله وظایف سیستم عامل سرویس دهی است در واقع سرویس‌های سیستم عامل‌های چندپردازنده بین چندپردازنده توزیع می‌شوند و یا هر پردازنده یک یا چند سرویس دارد.

۱- رئیس / مرئوس (مشتری / خدمتگزار (master slave)): در این روش هسته بر روی یک پردازنده قرار می‌گیرد و این پردازنده فرایندهای موجود را بر روی پردازنده‌های دیگر توزیع می‌کند فرایندها به صورت موازی انجام می‌شوند در نتیجه سرعت اجرا بالا می‌رود، زمان اجرا کاهش می‌یابد.



ویژگی‌های روش master/slave: ۱- پیاده‌سازی و مدیریت آن بسیار ساده است.

۲- قابلیت اطمینان پایین (با از کار افتادن یک پردازنده (پردازنده مربوط به هسته))

CACTUS

Scanned by CamScanner

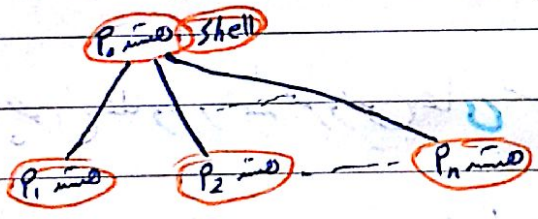
Year:..... Month:..... Day:.....

Subject:.....

فرایندهای اجرایی دیگر از کار می افتند. ۲- وجود گلوگاه: پدیده‌ای که در آن چند پردازنده برای اجرای وظایف خود می خواهند از یک منبع (پردازنده) استفاده کنند.

۴- کاهش سرعت: برای فرایندهایی که نیاز به فراخوانی سیستمی (systemcall) دارند.

۱- سرپرست جزا: در این حالت هسته سیستم عامل بر روی تمام پردازنده‌ها بصورت جداگانه قرار می گیرد. اما تنها یک shell (پوسته ارتباط با کاربر) وجود دارد که فرایندهای دیگر هم برای ارتباط با کاربر از همین پوسته استفاده می کنند. هسته بر روی تمام پردازنده‌ها قرار می گیرد و روی هر پردازنده چندین فرایند می تواند اجرا شود.



ویژگی های روش سرپرست جزا: این روش مشکلات روش master/slav ندارد

۲- تکرار هسته سیستم عامل فضای زیادی از سیستم عامل اشغال می کند. مدیریت پیچیده‌تری دارد.

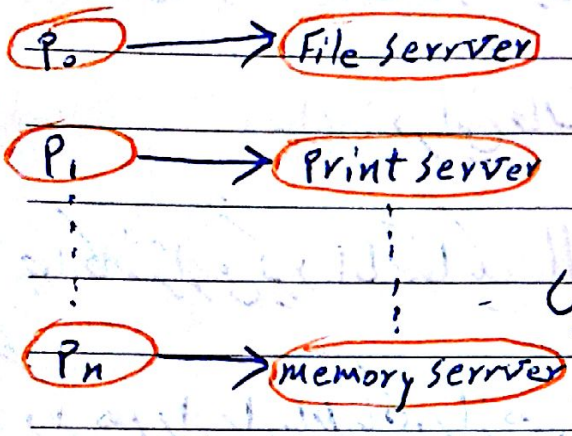
۳- اگر پوسته یا پردازنده دارای پوسته از کار بیفتد امکان ارتباط فرایندها با کاربر قطع شود هرچند که پردازنده‌ها می توانند به کار پردازشی خود ادامه دهند ۴- افزایش سرعت: برای

Scanned by CamScanner

Year:..... Month:..... Day:..... Subject:.....

فرایندهایی که نیاز به system call دارند هرزمانی که نیاز باشد همزمان ایجاد می شود زیرا لایه های سیستم عامل برروی تمام پردازنده ها قرار دارد.

روش سرریست تنها: این روش ترکیبی از دو روش قبل است ریزهسته سیستم عامل برروی تمام پردازنده ها قرار دارد هریک از سرویس های سیستم عامل برروی یکی از پردازنده ها قرار می گیرد.



ویژگی های روش فوق: نسبت به روش master/slave جای بیشتری اشغال می کند چون علاوه بر هسته سرویس های سیستم عامل هر پردازنده یک ریزهسته نیز دارد. ۲- نسبت به روش سرریست مجزا فضای کمتری اشغال می کند چون سرویس ها توزیع شده اند ۳- تا حدودی از اجرای هم زمان و موازی فرایندها استفاده می کند.

CACTUS
Scanned by CamScanner

الگوریتم های زمانبندی ایستا در چند پردازنده ای: الگوریتم های زمانبندی قابل پس گرفتنی: مثل Bin packing ، الگوریتم های زمانبندی غیرقابل پس گرفتنی مثل: SPT و LPT، RLPT، SLPT

معیارهای زمانبندی: ۱- میزان موازی سازی

$$\text{میزان موازی سازی} = \frac{\text{جمع کل زمان های اجرا}}{\text{زمان اجرای کل}}$$

$$\text{بهره وری CPU} = \frac{\text{زمان پردازش}}{\text{زمان کل}}$$

$$\text{توان عملیاتی} = \frac{\text{تعداد کل کارهای انجام شده در واحد زمان}}{\text{زمان کل}}$$

الگوریتم زمانبندی ایستا قابل پس گرفتنی Bin packing: یک الگوریتم قابل پس گرفتنیست: در این روش به هر پردازنده حداکثر زمانی به اندازه FTopt داده می شود.

$$FTopt = \max\left\{\frac{\sum T_i}{m}, \max(T_i)\right\}$$

$\frac{\sum T_i}{m}$: مجموع طول کارها / تعداد پردازنده ها

$\max(T_i)$: طول بزرگترین کارها

Scanned by CamScanner

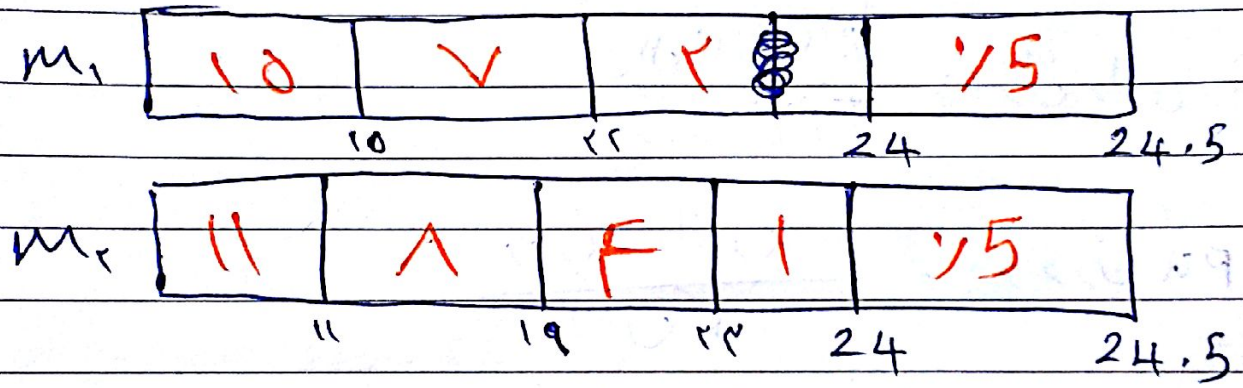
Year:..... Month:..... Day:..... Subject:.....

مثال: زمان کارهای موجود در سیستم عبارت اند از $T_i = \{15, 11, 8, 7, 4, 2, 1, 1\}$

$m = 2$

$$FT_{opt} = \max\left\{\frac{15+11+8+7+4+2+1+1}{2}, 15\right\}$$

$$FT_{opt} = \max\{24.5, 15\} = 24.5$$



یک الگوریتم غیرقابل پیش گرفتنی است: SPT (shortest processing Time)

کارها را در این الگوریتم براساس زمان اجرایشان بصورت صعودی مرتب می کنیم. کوچکترین کار را در اولین پردازنده آزاد قرار می دهیم و کار بعدی را در پردازنده آزاد بعدی قرار می دهیم الی آخر تا تمام کارها زمانبندی شوند.

Scanned by CamScanner

Year:..... Month:..... Day:..... Subject:.....

مثال: زمان کل موجود در یک سیستم با ۴ پردازنده عبارت اند از:

$T_i = \{1, 4, 7, 7, 9, 15, 18, 20, 30, 70\}$ $m = 4$

M_1: | 1 | 7 | 18 | 70 |
1 8 26 96

M_2: | 4 | 9 | 20 |
4 13 33

TfT = 96

M_3: | 7 | 15 | 30 |
7 22 52

الگوریتم LPT همانند الگوریتم SPT می باشد با این تفاوت که بصورت نزولی مرتب می کنیم و اولین بار بزرگترین را انتخاب می کنیم. برعکس SPT که کوچکترین را انتخاب می کنیم. زمان انتظار از SPT بهتر است ولی در کل نسبت به SPT بهتر است. زمان TfT را کاهش می دهد.

LPT (Largest processing Time)
Longest . . . .

$T_i = \{70, 30, 20, 18, 15, 9, 7, 7, 4, 1\}$

M_1: | 70 |

M_2: | 30 | 15 | 7 | 4 |
30 45 52 56

M_3: | 20 | 18 | 9 | 7 | 1 |
20 38 47 54 55

TfT = 70

CACTUS

Scanned by CamScanner

Year:..... Month:..... Day:..... Subject:.....

RLPT: یک الگوریتم غیرقابل پیش گرفتنی است

کارها را براساس زمان اجرایشان بصورت نزولی مرتب می‌کنیم، بزرگترین کار را در اولین پردازنده آزاد قرار می‌گیرد کار بعدی در پردازنده آزاد بعدی و الی آخر تا کارها زمانبندی شود برای هر پردازنده لیست کارهای اجرایی معکوس می‌شود.

این الگوریتم با TFT الگوریتم LPT برابر است

و اما زمان اجرا یا پاسخ بهتری تولید می‌کند و عملی هم هست:

مثال قبل:

M_1: 7o

M_2: 4 | 7 | 15 | 30

4 11 26 56

M_3: 1 | 7 | 9 | 18 | 20

1 8 17 35 55

TFT = 7o

TFT: یکی از معیارهای زمانبندی است در سیستم های چند پردازنده‌ای است

حداکثر زمانی که برای اجرای کل کارها طول می‌کشد را TFT می‌گویند

CACTUS

Scanned by CamScanner

Year:..... Month:..... Day:..... Subject:.....

LSPT یا SLPT: LSPT یک الگوریتم غیرقابل پیش گرفتنی است که ترکیبی از الگوریتم های SPT و LPT است کارها را براساس زمان اجرایشان بصورت صعودی مرتب می کند سپس به ترتیب کارها را به دسته های m تایی تقسیم می کنیم و به ترتیب mتا ها بصورت LPT زمانبندی می شود یعنی بزرگترین کار در اولین پردازنده آزاد قرار می گیرد و کار بعدی در پردازنده آزاد بعدی و الی آخر تا تمام mتا ها زمان بندی شوند.

$T_i = \{1, 4, 7, 7, 9, 15, 18, 25, 30, 70\}$ $m=3$

M_1: | 7 | 7 | 20 | 70 |
0 — 7 — 14 — 34 — 104

M_2: | 4 | 9 | 30 |
0 — 4 — 13 — 43

M_3: | 1 | 15 | 18 |
0 — 1 — 16 — 34

TFT = 104

Scanned by CamScanner

Year:..... Month:..... Day:.....

Subject:.....

الگوریتم های زمان بندی پردازه در سیستم های چند پردازنده ای: ۱- استراک بار ۲- زمان بندی گروهی ۳- اختصاص پردازنده به فرایند ۴- جداسازی سیاست از مکانیزم زمان بندی.

۱- استراک بار (الوجه ای): فرایندها وارد صف مشترک مابین پردازنده ها می شوند پردازنده های بیکار آن ها را از صف برمی دارند و اجرا می کنند. بعد از تمام شدن زمان در نظر گرفته شده برای هر یک از فرایندها (زمان بندی قابل پس گرفتنی) یا اتمام فرایند یا انجام عمل I/O (زمان بندی غیر قابل پس گرفتنی) فرایند دوباره به انتهای صف می رود و پردازنده ای که از انجام شدن فرایندش آزاد شده فرایند بعدی را از صف برای اجرا شدن برداشت.

ویژگی این الگوریتم: همیشه پردازنده ها بار متوازن در حال کار کردن هستن و معمولا پردازنده بیکار نداریم. صف مشترک هرگونه می توان پیاده سازی نمود و به روش های مختلف زمان بندی کرد. مدیریت صف مشترک ساده است. صف مشترک بعضی وقتا ممکن است باعث ایجاد گلوگاه شود.

CACTUS

Scanned by CamScanner

Year:..... Month:..... Day:.....

Subject:.....

۲. الگوریتم زمان‌بندی گروهی: گروهی از فرایندها که با هم ارتباط بیشتری دارند یا یک فرایند به همراه نخ‌های آن با یکدیگر زمان‌بندی شده و هم زمان به گروهی از پردازنده‌ها تخصیص داده شده و یا از گروه گرفته می‌شوند.

۳. الگوریتم اختصاص پردازنده به فرایند: گروهی از فرایندها که با هم ارتباط بیشتری دارند یا یک فرایند به همراه نخ‌های آن به گروهی از پردازنده‌ها تخصیص داده شده‌اند، هیچ گاه پردازنده از فرایند پس گرفته نمی‌شود مگر اینکه فرایند تمام شود، سرعت بالایی دارد چون تعویض فرایند نداریم، عیب بزرگ این روش بیکار بودن پردازنده و کاهش بهره وری پردازنده است.

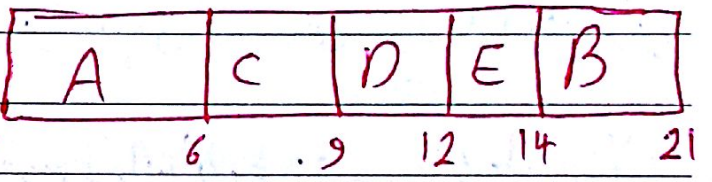
۴. جداسازی سیاست از مکانیزم زمان‌بندی: یک ایده کلی است و ربطی به زمان‌بندی ندارد. مکانیزم طریقه پیاده‌سازی سیاست است. مثلاً سیاست زمان‌بندی را مشخص می‌کند که زمان‌بندی بصورت ایستا باشد یا پویا. اما مکانیزم آن مشخص می‌کند برای سیاست در نظر گرفته شده از چه روشی استفاده می‌شود.

Year:..... Month:..... Day:.....

Subject:.....

سوال: یک سیستم تک پردازنده‌ای با زمان بالاترین نسبت پاسخ HRN را در نظر بگیرید درصورتی که فرایندها با زمان‌های اجرای ۶ و ۷ و ۳ و ۳ و ۲ و ۲ به ترتیب در زمان‌های ۰ و ۱ و ۳ و ۵ و ۷ وارد سیستم شوند کدام فرایند آخرین فرایند اجرایی خواهد بود.

فرایندها	زمان اجرا	زمان ورود
A	6	0
B	7	1
C	3	3
D	3	5
E	2	7



$$\text{اولویت} = \frac{\text{زمان سرویس} + \text{زمان انتظار}}{\text{زمان سرویس}}$$

CACTUS

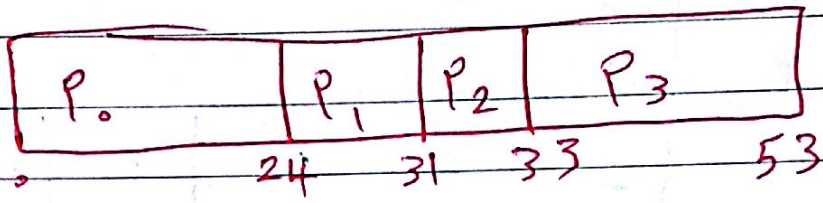
Scanned by CamScanner

Year:..... Month:..... Day:.....

Subject:.....

Fcfs: فرایندهای زیر را در نظر بگیرید و با استفاده از زمان بندی Fcfs متوسط زمان پاسخ را بدست آورید؟

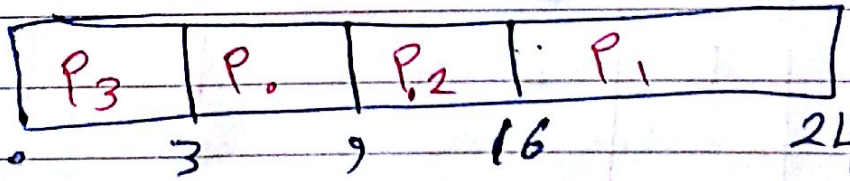
فرایندها	زمان اجرا
P_0	24
P_1	7
P_2	2
P_3	20



متوسط زمان پاسخ $= \frac{24+31+33+53}{4} = 35.2$

sjf: فرایندهای زیر را در نظر بگیرید و با استفاده از زمان بندی sjf (انحصاری) متوسط زمان انتظار را حساب کنید؟

فرایندها	زمان اجرا
P_0	6
P_1	8
P_2	7
P_3	3



زمان انتظار:

$P_0 = 9 - 0 - 6 = 3$

$P_1 = 24 - 0 - 8 = 16$

$P_2 = 16 - 0 - 7 = 9$

$P_3 = 3 - 0 - 3 = 0$

Scanned by CamScanner

Year:..... Month:..... Day:.....

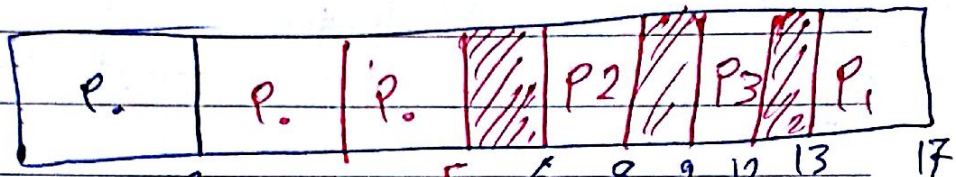
Subject:.....

SJF (پیش‌دستانه)

سیستم عاملی از روش SJF (انحصاری) استفاده می‌کند.

متوسط زمان انتظار برای چهار فرایند زیر را محاسبه کنید؟

فرایندها	زمان اجرا	زمان ورود
P_0	5	0
P_1	4	2
P_2	2	4
P_3	3	6



$P_0 = 3$, $P_1 = 4$

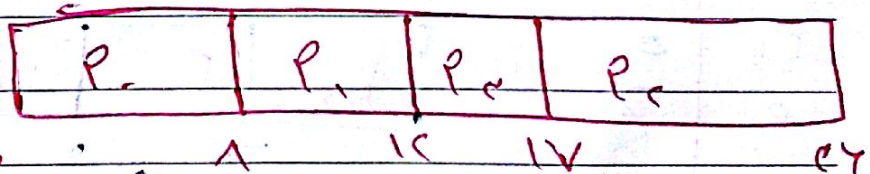
$P_0 = 1$, $P_1 = 4$, $P_2 = 2$

$P_1 = 4$, $P_2 = 2$, $P_3 = 3$

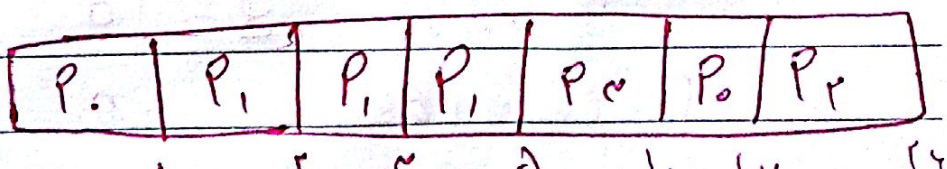
$P_1 = 4$, $P_3 = 3$

SJF , SRT

فرایندها	زمان اجرا	زمان ورود
P_0	8	0
P_1	4	1
P_2	9	2
P_3	5	3



SJF



CACTUS

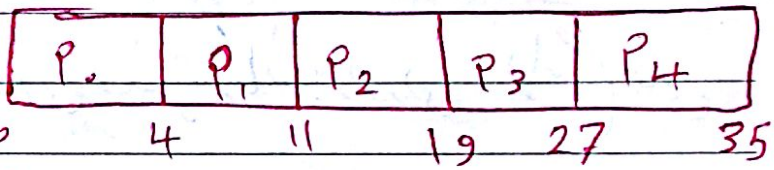
Scanned by CamScanner

Year:..... Month:..... Day:..... Subject:.....

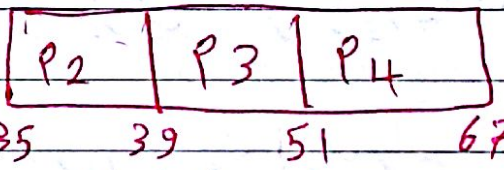
MLFQ: یک سیستم با الگوریتم سه سطحی را در نظر بگیرید. اگر صف اول (سطح اول) از الگوریتم RR با کوانتوم زمانی ۸ms و صف دوم از الگوریتم RR با کوانتوم زمانی ۱۶ms و صف سوم با الگوریتم FiFo زمان بندی شود. با توجه به جدول زیر میانگین زمان بازگشت و میانگین زمان انتظار را حساب کنید؟

فرایندها	زمان اجرا
P_0	4
P_1	7
P_2	12
P_3	20
P_4	30

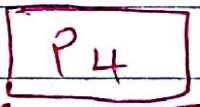
کوانتوم = 8 →



کوانتوم = 16 →



FiFo →



در یک سیستم انحصاری از الگوریتم اولویت استفاده می‌شود فرایندهای زیر را در نظر بگیرید متوسط زمان انتظار، اندازه زمان CPU، زمان پاسخ و زمان عملیاتی را حساب کنید؟ اولویت با عدد کوچک است و زمان ورود فرایندها را صفر در نظر بگیرید.

CACTUS

Scanned by CamScanner

Year:..... Month:..... Day:.....

Subject:.....

فرایندها	زمان اجرا	اولویت
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

۹۴۱,۲۲

مثال: سیستم عاملی از الگوریتم زمان بندی صف چندگانه بازخوردی با کوانتوم های مختلف به صورت $q_k = 2^{i-1}$ استفاده می کند. اگر فرایندها ابتدا وارد صف اول با کوانتوم زمانی 1 میلی ثانیه شوند برای اجرای کامل یک فرایند با زمان اجرای 5 میلی ثانیه با بیستی وارد چند صفحه مختلف شوند؟

$q_k = 2^{i-1}$

$q_1 = 2^{1-1} = 1$

$q_2 = 2^{2-1} = 2$

$q_3 = 2^{3-1} = 4$

صف ۱ | 1

صف ۲ | 2

صف ۳ | 2

CACTUS

Scanned by CamScanner

Year:..... Month:..... Day:..... Subject:.....

همزمانی فرآیندها:

دو فرایند یا مستقل از هم هستند یا همکاری با هم دارند. فرایندهای مستقل هیچ اثری بر هم ندارند و به هر ترتیبی که اجرا شوند خروجی نهایی هر دو همواره یکسان است. ولی دو فرایند همکار بر هم اثر دارند و اینکه کدام زودتر و به چه ترتیبی و در چه زمان هایی بین آن دو سوئیچ کند در ایجاد پاسخ نهایی تأثیر دارد.

مثال: دو فرایند P_1 و P_2 بصورت هم روند اجرایی شوند و امکان اجرای آنها بصورت interleaved وجود دارد.

(interleaved: به این معناست که در هر زمان امکان تعویض متن به فرایند دیگر وجود دارد) در صورتی که مقدار اولیه متغیر سراسری A صفر باشد بعد از اجرای کامل دو فرایند مقادیر A, B, C چه خواهد شد.

$a = 0$

P_1 کد	P_2 کد
$a = 1$	$b = a$
	$c = a$

اجرای کامل	اجرای (۲)	اجرای (۳)
$P_1 \rightarrow a = 1$	$P_2 \begin{cases} b = 0 \\ c = 0 \end{cases}$	$b = 0$
$P_2 \rightarrow \begin{cases} b = 1 \\ c = 1 \end{cases}$	$P_1 \Leftrightarrow a = 1$	$a = 1$
		$c = 1$

CACTUS

Scanned by CamScanner

Year:..... Month:..... Day:..... Subject:.....

مثال: فرض کنید مقدار شمارنده c در حال حاضر برابر 5 باشد و پردازش های تولید کننده و مصرف کننده c:=c+1 و c:=c-1 را هر زمان اجرا کنند. پس از اجرای متغیر c ممکن است 4 یا 5 یا 6 شود. ولی نتیجه درست فقط مقدار 5 می باشد. روبنه زیر نمونه از دستورات c می باشد.

```
c := c+1
mov Ax, c        Ax ← 5
Add Ax, 1        Ax ← 6
mov c, Ax        c ← 6
```

```
c := c-1 →
mov Bx, c →      Bx ← 6
sub Bx, 1        Bx ← 5
mov c, Bx        c = 5
```

همانی که CPU بین فرایندهای تولید کننده و مصرف کننده سویچ می کند ممکن است ترتیب دستورات زیر اجرا شود. نهایتا متغیر c برابر مقدار نادرست 4 می شود. دلیل رسیدن به این جواب نادرست این است که اجازه داده ایم هر دو فرایند

Scanned by CamScanner

Year:..... Month:..... Day:..... Subject:.....

متغیر c بطور هم زمان دسترسی داشته باشد. برای جلوگیری از این مشکل باید کاری کنیم که تنها یک پردازش در یک زمان بتواند متغیر مشترک را دستکاری کند.

جهت این تضمین به نوعی هم زمانی فرایندها نیاز داریم.

ناصیه بحرانی: ناحیه‌ای که در آن ۲ یا چند فرایند برروی داده‌های اشتراکی کار می‌کنند و چون دسترسی هم زمان فرایندها به داده‌های اشتراکی باعث خطا می‌شود. برای جلوگیری از این خطا مفهوم دسترسی انحصاری (انحصاری متقابل) مطرح می‌شود.

مسئله

انحصار متقابل یعنی در هر زمان فقط یک فرایند می‌تواند به داده‌ها اشتراکی دسترسی داشته باشد.

۵ راه حل برای مشکل ناصیه بحرانی:

۱- راه حل نرم افزاری ۲- راه حل سخت افزاری ۳- راه حل سیستم عامل و ابزارهای همزمانی ۴- حمایت ساختاری زبان برنامه نویسی ۵- تبادل پیام

یک راه حل برای مشکل ناصیه بحرانی باید ۳ نیاز زیر را برآورده نماید:

۱- دو به دو ناسازگاری (انحصار متقابل): اگر فرایندی در حال اجرا در ناصیه بحرانی خود باشد در آن مدت هیچ فرایندی نمی‌تواند در ناصیه بحرانی اش اجرا شود

CACTUS

Scanned by CamScanner

Year:..... Month:..... Day:..... Subject:.....

۲- شرط:

اگر هیچ فرایندی در ناحیه بحرانی خود نباشد و فرایندهای دیگر تمایل به ورود به ناحیه بحرانی خود را داشته باشند، فقط آن فرایندهایی که هنوز به بخش بحرانی نرسیده اند می توانند در تصمیم گیری درباره اینکه کدام فرایندها به ناحیه بحرانی خود وارد شوند شرکت کنند.

۳- انتظار محدود:

برای تعداد دفعاتی که به فرایندهای دیگر اجازه داده می شود وارد ناحیه بحرانی خود شوند باید حدی وجود داشته باشد. این محدودیت باید بداند از اینکه یک فرایند تقاضایی برای ورود به ناحیه بحرانی اش را درخواست کرد تا قبل از پذیرش آن درخواست انجام شود.

CACTUS

Scanned by CamScanner

Year:..... Month:..... Day:.....

Subject:.....

راه حل های نرم افزاری:

اولین تلاش:

```
int turn = 0;

P0
:
while (turn != 0)
    /* do nothing */;
    /*  ناصر برای */;
    turn = 1;
:

P1
:
while (turn != 1)
    /* do nothing */;
    /*  ناصر برای */
    turn = 0;
:
```


Scanned by CamScanner

Year:..... Month:..... Day:..... Subject:.....

اولین تلاش! فرآیندی که می‌خواهد بخش بحرانی خود را اجرا کند ابتدا مقدار turn را بررسی می‌کند. اگر مقدار turn مساوی شماره‌ی این فرآیند باشد در این صورت این فرایند وارد ناحیه بحرانی خود می‌شود. در غیر اینصورت مجبور است صبر کند. فرایند منتظر عمل خواندن مقدار turn را آنقدر تکرار می‌کند تا مجاز به ورود به ناحیه بحرانی خود شود این رویه را انتظار مشغول می‌نامیم. بنابراین هنگامی که این فرایند منتظر فرصت خود می‌باشد، مشغول است.

* این خاصیت انحصار متقابل را تضمین می‌کند اما ۲ تا مشکل دارد:

۱- فرآیندها در استفاده از بخش‌های بحرانی خود باید دقیقاً متناوب و یکی در میان عمل کنند. بنابراین سرعت اجرا به وسیله فرآیند کندتر هدایت می‌شود.

۲- اگر فرایندی با شکست مواجه شود چه در بخش بحرانی و چه در خارج بخش بحرانی در هر حال فرآیند دیگر را مسدود می‌کند.

CACTUS

Scanned by CamScanner