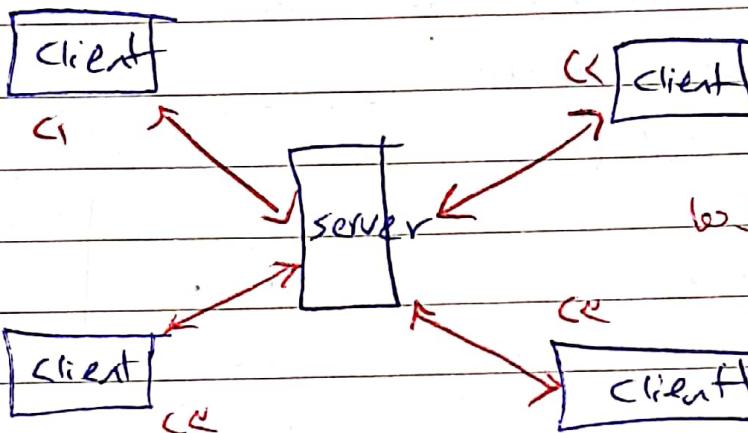


SESSION 8

صلال و خرفنی کن سے لور داریم کہ یہ طب بجھنیں کلائنس پاسخ

کلائنس و مفہوم و رخوبیوں کا لور بدل لئے
کہ فائل نہ رکھ را بفرماد
لور صفوی اسی درخواست کی کلائنس کو صفتی کرنا



صلال جوہ و

لور خانہ بے کلائنس دعا
یہ طب بایخ رکھ

```

    while (true) {
        ایک (یہ کسی thread کے لئے)
        c1 = accept()
        دوسرے کلائنٹ کیلئے
        read (c1, fname) ←
        رام خانہ
        for (size fik) {
            (رکھ لے بارے کی تعلیم
            کلائنٹ
        }
        یہ thread
    }

```

صلال اسی نوع ایسا تو ہے

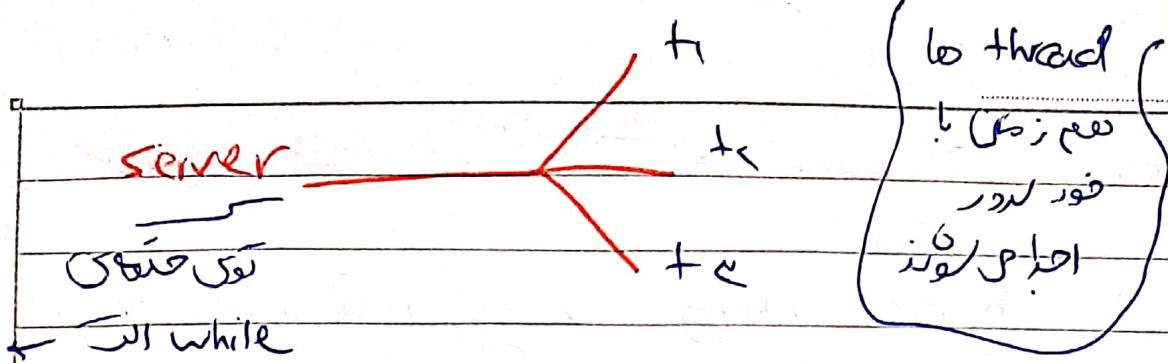
تو یہ ایسا لور مسکر کرائے تو اسی خانہ رکھوںدے و بھی
لور باید بے نازدہ کل میتھی فائل و سوت اسکا کلائنس

باید حکم کر کر لور تا لور بتوونے میں → نعم زمانی ازس

کلائنٹ دیکھ کر کلائنس دیکھ کر

hirmandpaper

کام جو اپ برسک



چون سرور را باید نوشت
multiprocess یا، چون سرور را باید نوشت
process یا threads باید نوشت

(Interprocess communication) IPC میان پردازنده های بین سرور و سرور
برقرار کردن ارتباط بین سرور و سرور ①
لور لور و لور لور ②

Motivation

برای multithread ← (سیستم application) برای
برای interactive کاربری که

برای application داخل یک thread
thread که در یک کامپیوتر باشند که task
word بینهم می‌باشد

spell check → fetch data → update display
برای کلائنٹ ها و کلائنٹ
{ answer a network
request

heavy weight) یعنی کارکرده یک process یعنی
memory space یعنی دارای

رازنده حافظه و overhead

light-weight) یعنی کارکرده یک thread

efficiency ↑ ← Intros. to thread (چیز که می‌خواهیم)

basic sorting

trees

الگوریتم های گراف

گراف

AI

مثال (کاربردی) حالت اولیه

کاربردی (کاربردی)

times CPU-intensive کارهای سنجاق

data mining کار

machine learning

CPU بروگران

پردازش multi core پردازش

چند نویس

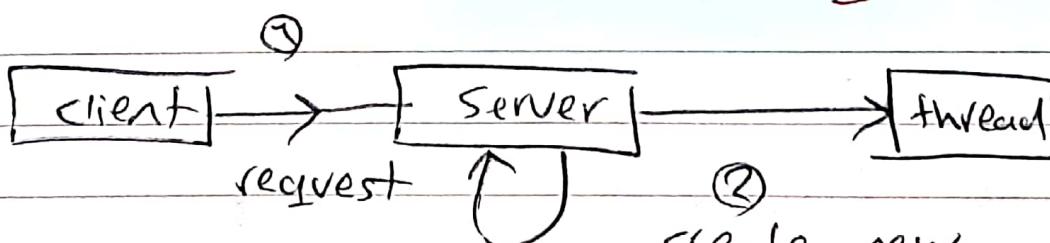
که برخاطرها طالق در بنائند و اینجا (در)

گونه که پردازش را کنی توانیم اینجاست

اینرا کسی که نبودن بودن core که اینرا

نمی‌تواند فایده‌ای ندارد. (اینی بالای بودن لردست)

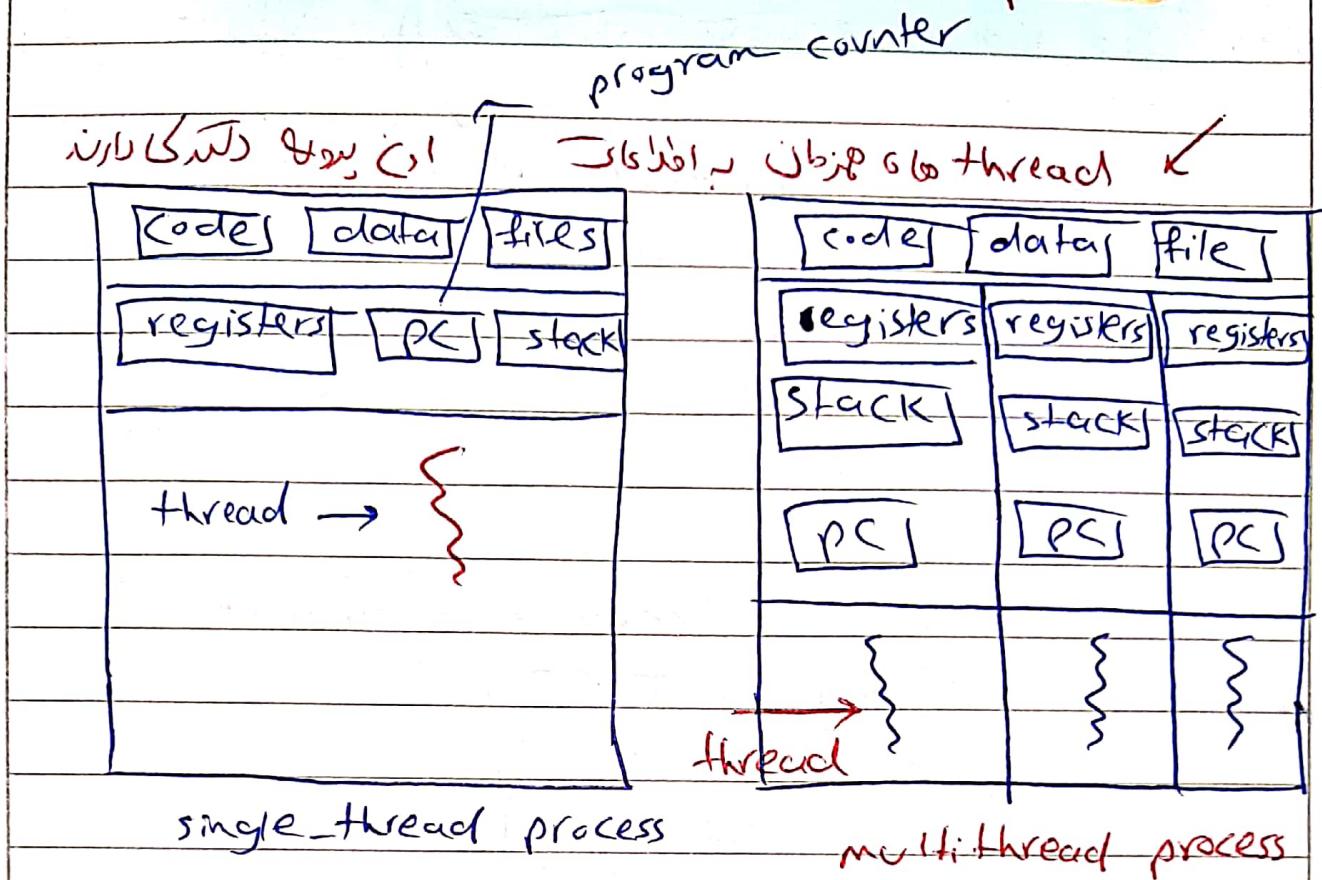
(IntelliJ IDEA (سیلو))



resume
listening for
additional client
requests

create new
thread to service
the request

Single & multithread processes



ویل کے file, data, code اور
ویل کے process کے file thread کے

register, stack, PC کے بارے میں
thread no ہے

ویل کے thread کے بارے میں
ویل کے CPU کے بارے میں
ویل کے process کے بارے میں
ویل کے context switch کے بارے میں
ویل کے process کے بارے میں

ویل کے overheard کے بارے میں
ویل کے thread کے بارے میں

ویل کے overhead کے بارے میں
ویل کے context switch کے بارے میں
ویل کے process کے بارے میں

اگر کسی بھی از روند ملک
وہ اور 6 اور 5 کی خوبی (زیاد)

responsiveness

pure user interface (UI) ←
وہ (time-consuming) میں کی

اور 6 asynchronous & FIFO thread

انسانی و Responsive پر بڑا 6 application

use of 6 thread ② → **resource sharing**

میں کی share کے لئے 6 میں کی share

پر 6 کے بارے میں دیکھو

حلقہ مارنے

6 میں کی share → shared memory (6 میں کی share)

message passing

shared memory 6 میں کی share

IPC (Inter Process Communication) 6 میں کی share

میں کی share 6 global value 6 thread

Economy

use of 6 thread ③ →

use of 6

(Single) process

6 thread (context switch) 6 multi process

use of 6

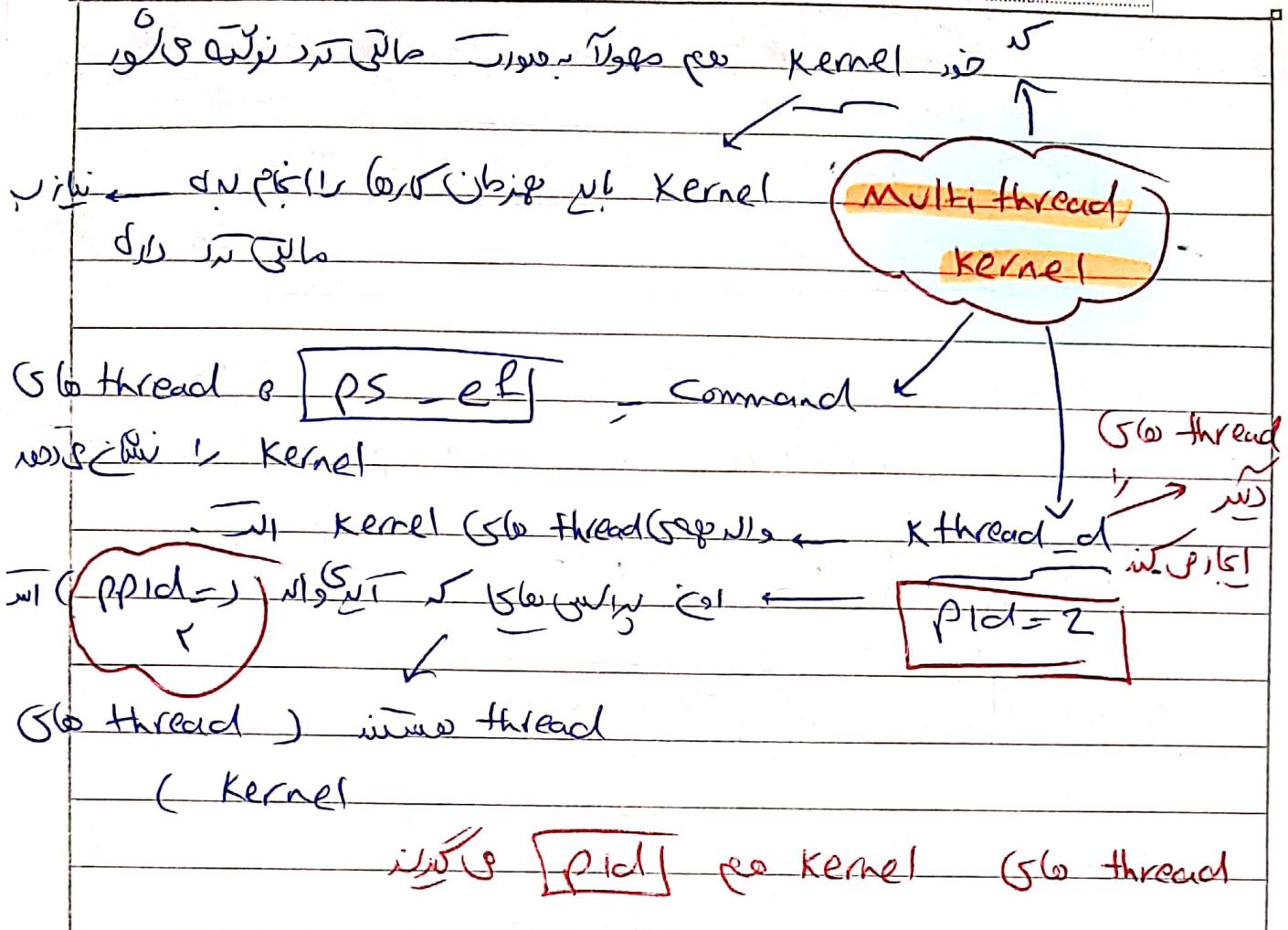
use of 6

scalability

use of 6

use of 6

use of 6



Concurrency vs parallelism

Supports more than one task & Concurrency
making process

Single processor/core & scheduler
providing Concurrency

Two types of parallelism ← parallelism
no Ptid & task

جیوپر سی پی یو

Concurrent

Single-core

جیوپر سی پی یو



Single core

time sharing

جیوپر سی پی یو کو اسکرین کرنا
کو اسکرین کرنا

multi core جیوپر سی پی یو پاراللیزم

Core 1

T_1

T_c

T_E

T_1

T_c

T_E

T_1

—

Core 2

T_c

T_E

T_1

T_c

T_E

T_c

—

جیوپر سی پی یو اسکرین کرنا

time

جیوپر سی پی یو اسکرین کرنا

Core 1

جیوپر سی پی یو اسکرین کرنا

Core 2

جیوپر سی پی یو اسکرین کرنا

جیوپر سی پی یو اسکرین کرنا

time sharing

جیوپر سی پی یو اسکرین کرنا

جیوپر سی پی یو اسکرین کرنا

hirmandpaper

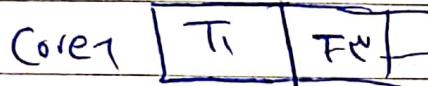
جیوپر سی پی یو اسکرین کرنا

جائز (عزم زمان) هستند (دسته رفاهی) \leftarrow Single core تری حسنه \rightarrow

واعقی واعقی

multi-core (دسته ترقیاتی) دسته زمان واعقی می توانند باشد \leftarrow multi-core (دسته ترقیاتی) دسته زمان واعقی

thread



تسلیم \leftarrow ترکیب T2, T1 \rightarrow که دسته زمان واعقی هستند



دسته زمان واعقی نسبت اما ما \leftarrow ترکیب T2, T1
بسیاری کم دسته زمان \rightarrow چون کار مسروچه کنی تو که CPU (درستی) ای

برای یک thread

multicore programming

اگر بی هم تری برداشت از التقاره کنی

(می خواهیم نکرد اجرای برنامه را برا برهم وای برنامه تواند

که کلی مختلف دسته زمان اجرای این Core (دسته

حفوری) که

بنویسید

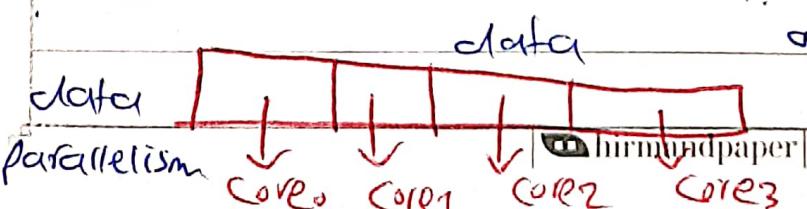
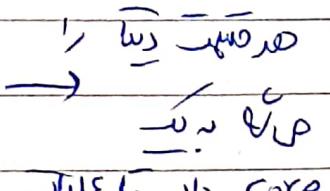
data parallelism

که دسته نیز کار کری داریم که این دسته را ب

کنی (کار) کلی می سازیم

تری (این حالت ای که قرار دادیم) task

که انجام شود، و میان رسانیده (کار) را ب



دسته زمان

کار را داریم

اینها انجام دارند

اللئارنینگ (Learning) machine learning چیزی بینایی گیری و دادا از
parallelism

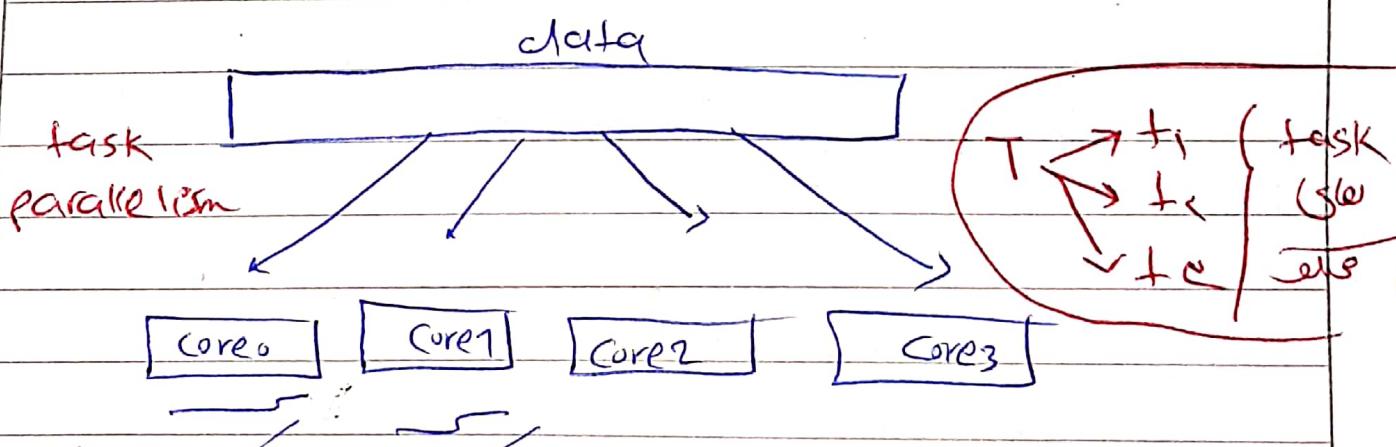
تعداد زیادی عیارات فریب فهم زمان ایام
که اینها را در یک مدت زمانی انجام داده اند

CPU کی کامل Core کی کامل Core \rightarrow GPU
ویژگی های فریب و فهم ایام (ویژگی های فرماتیکی)
عیاراتی فریب و فهم که خوبی نیز که مسند را فهم زمانی دارد
ویژگی های اینها از یک GPU کی کامل Core امداد کرد.

task های زیادی داشت که بین کارخانه نهاده شوند \leftarrow task parallelism

کارخانه های مختلفی داشت که task 6 Core را داشتند
ایامی کارخانه های مختلفی داشتند

کارخانه های مختلفی که task 6 Core داشتند



hybrid (hybrid) \leftarrow اینکارخانه ای دارد که دادا و فرماتیکی را دارد

data کی کمی کیلئے بھی توںیں بولیں Dividing activities
کوہا task اس کام کی task لئی کروں کیا
موہری لئی داری و کام نہ مارنے

task ایک ایسا اجنبیان اس کام کی task لئی کروں
Balance new tasks اس کام کی task لئی کروں
value

task کو تھاکر کر کر کوہا task لئی کروں Data splitting
کوہا task کو ایک ایسا task کی رائی کی کام کی task اس کام کی task
کوہا task کو ایک ایسا task کی رائی کی کام کی task اس کام کی task
کوہا task کو ایک ایسا task کی رائی کی کام کی task اس کام کی task

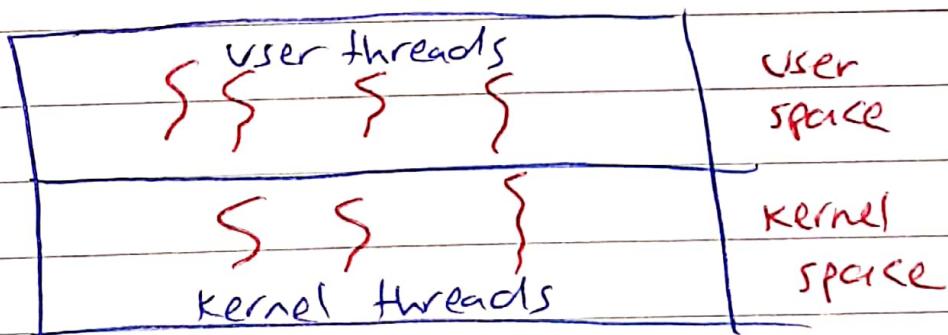
task کو ایک task کی رائی کی کام کی task اس کام کی task
کوہا task کو ایک task کی رائی کی کام کی task اس کام کی task
کوہا task کو ایک task کی رائی کی کام کی task اس کام کی task
کوہا task کو ایک task کی رائی کی کام کی task اس کام کی task
Testing and debugging

کوہا Core کوہا task کی رائی کی کام کی task اس کام کی task
کوہا task کو ایک task کی رائی کی کام کی task اس کام کی task
user threads
کوہا task کی رائی کی کام کی task اس کام کی task
کوہا task کی رائی کی کام کی task اس کام کی task
kernel threads
کوہا task کی رائی کی کام کی task اس کام کی task
کوہا task کی رائی کی کام کی task اس کام کی task
library threads
کوہا task کی رائی کی کام کی task اس کام کی task
کوہا task کی رائی کی کام کی task اس کام کی task
user threads
کوہا task کی رائی کی کام کی task اس کام کی task
کوہا task کی رائی کی کام کی task اس کام کی task
thread library
کوہا task کی رائی کی کام کی task اس کام کی task
کوہا task کی رائی کی کام کی task اس کام کی task
thread
کوہا task کی رائی کی کام کی task اس کام کی task
کوہا task کی رائی کی کام کی task اس کام کی task
thread

G, jisew 50 thread (for 80% v
 (000) level, jisew → POSIX pthreads ①
 Kernel level
 Kernel & 50+ thread windows threads ②
 java threads ③

is P types Kernel b/w → **Kernel threads**
 general purpose (500 OS) →
 virtually
 windows, solaris, linux, Tru64 Unix,
 Mac OS X

P Jisew Kernel & jisew 50 thread (jwbw)



many to one
 one to one
 many to many
multi thread
models

ج) thread یعنی چیزی که یک thread یا (سینکریونیتی) یعنی map از kernel

many to one

که block از thread یعنی چیزی که یک thread

که

پس

(5) parallel jobs یعنی سیستم یک thread

که یعنی چیزی که multicore یا چیزی که

که یعنی چیزی که thread یعنی چیزی که

که kernel

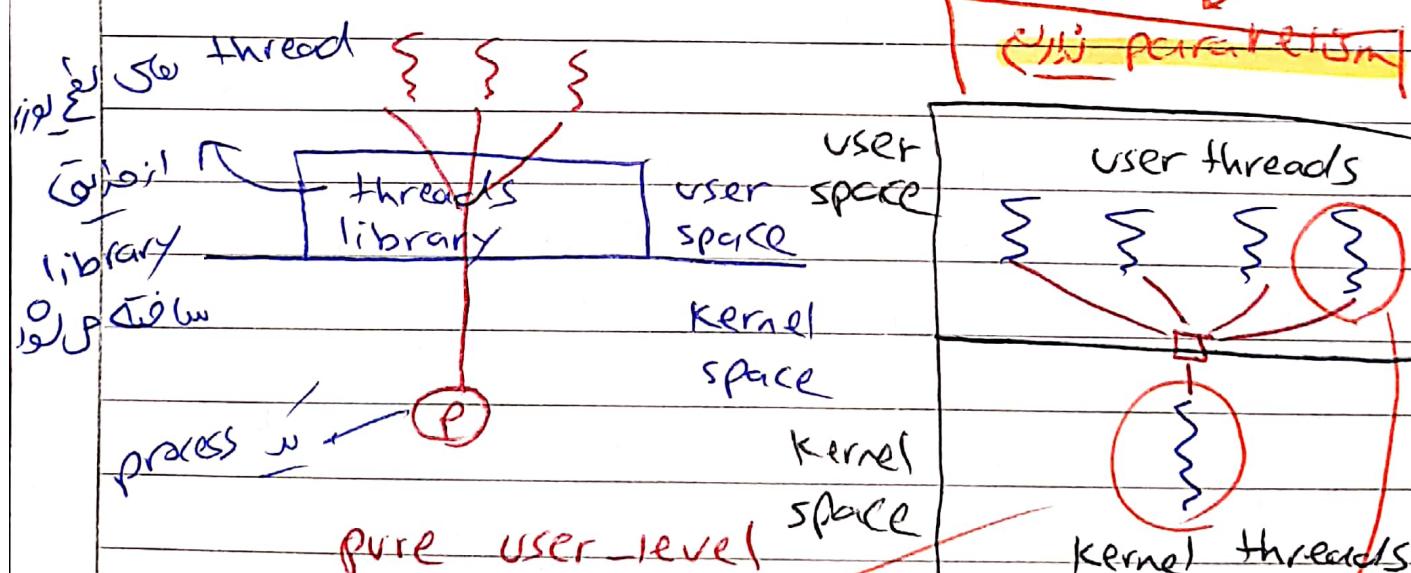
یعنی اینکه اینکه اینکه

Concurrency

① Solaris Green threads & jobs

که

② GNU portable threads



که ای دیگر چیزی که ای دیگر چیزی

که ای دیگر چیزی که ای دیگر چیزی

که

thread

each user-level thread maps to kernel-level thread

(one-to-one)

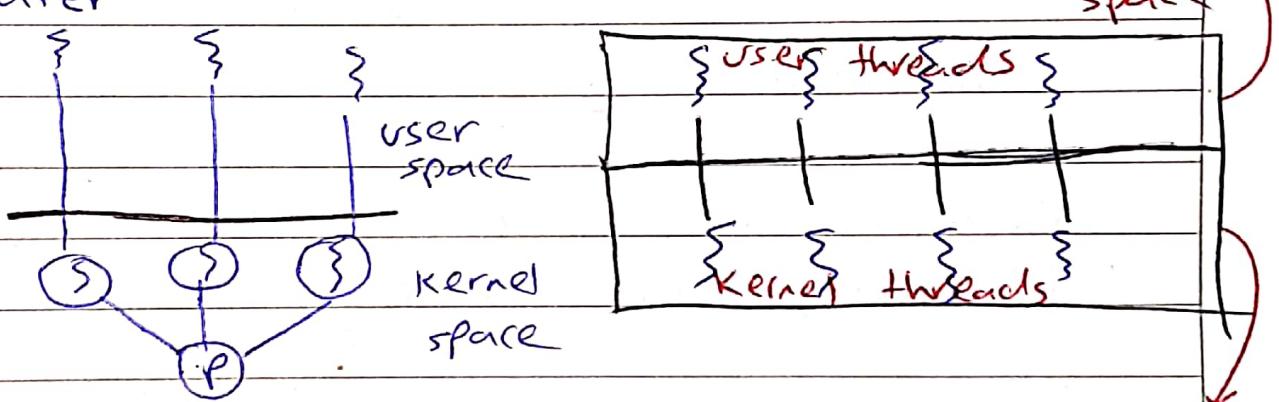
↳ thread ↳ ↳ thread ↳ user

↳ kernel

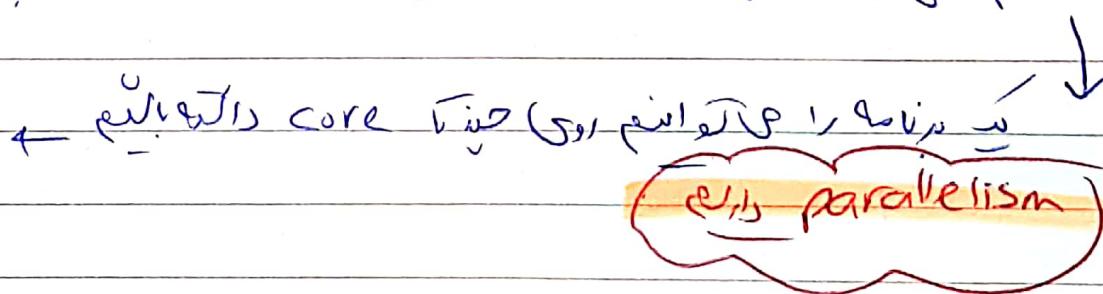
many-to-one ↳ concurrency

processes → threads → threads → overhead ↳ user parallelism

Solaris → Linux → Windows 8 later



سیستم پرداز کرنل می‌تواند چندین thread را در یک core اجرا کند
این روش بعدها سیستم پرداز را بسیار سریع می‌کند



پس کرنل می‌تواند چندین thread را در یک core اجرا کند

این روش بعدها سیستم پرداز را بسیار سریع می‌کند

int main() 6 thread را در یک core اجرا کند

برای اینجا از thread overhead استفاده نمی‌شود

Scanned by CamScanner

جولہ کی نوکی میں جو کام کرے جائے گی اسے overhead کہا جاتا ہے
 اس کا کام کیوں کرے جائے گی اسے thread (ٹھرڈ) کہا جاتا ہے
 اس کا کام کیوں کرے جائے گی اسے kernel (کرنل) کہا جاتا ہے

(Go threads) (Go threads)
 kernel user

Kernel ہے - thread (ٹھرڈ) کے مقابلے میں اسے OS کہا جاتا ہے
 جیسے

(Go thread) کے مقابلے میں اسے developer کہا جاتا ہے

Thread ہے کہ کام کرے جائے گی اسے kernel (Go thread) کہا جاتا ہے اسے developer کہا جاتا ہے

ایک Blocking - system call کے بعد thread کو وقت میں اسے وقت میں

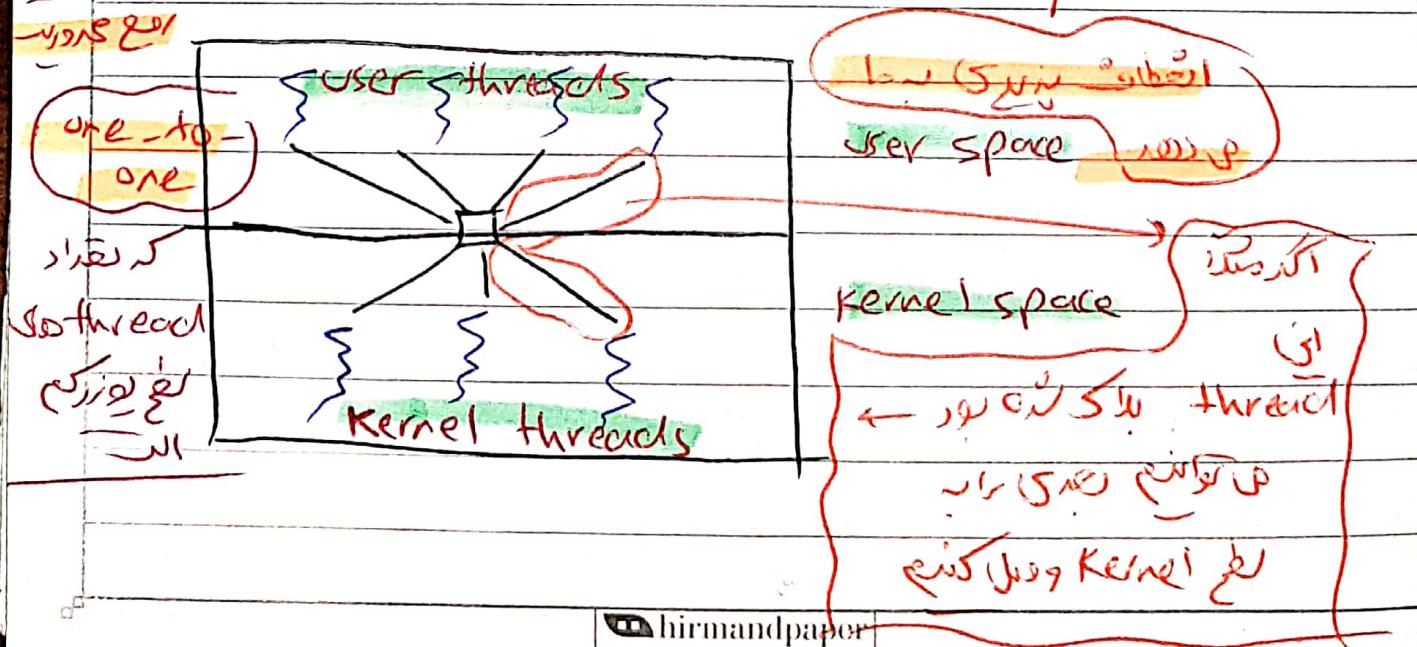
کام کرنے کے لئے کام کرنے کے لئے کام کرنے کے لئے کام کرنے کے لئے

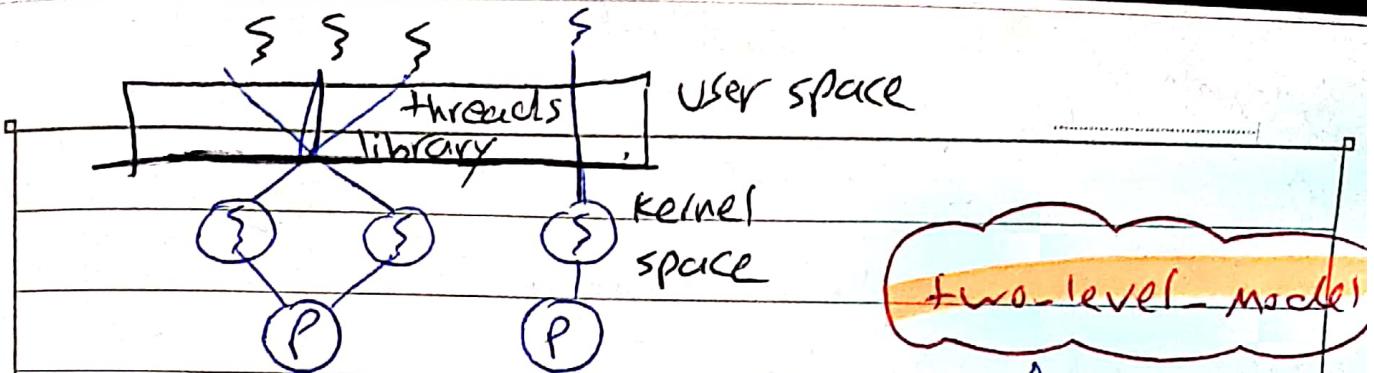
کام کرنے کے لئے کام کرنے کے لئے کام کرنے کے لئے کام کرنے کے لئے

کام کرنے کے لئے کام کرنے کے لئے کام کرنے کے لئے کام کرنے کے لئے

کام کرنے کے لئے کام کرنے کے لئے کام کرنے کے لئے کام کرنے کے لئے

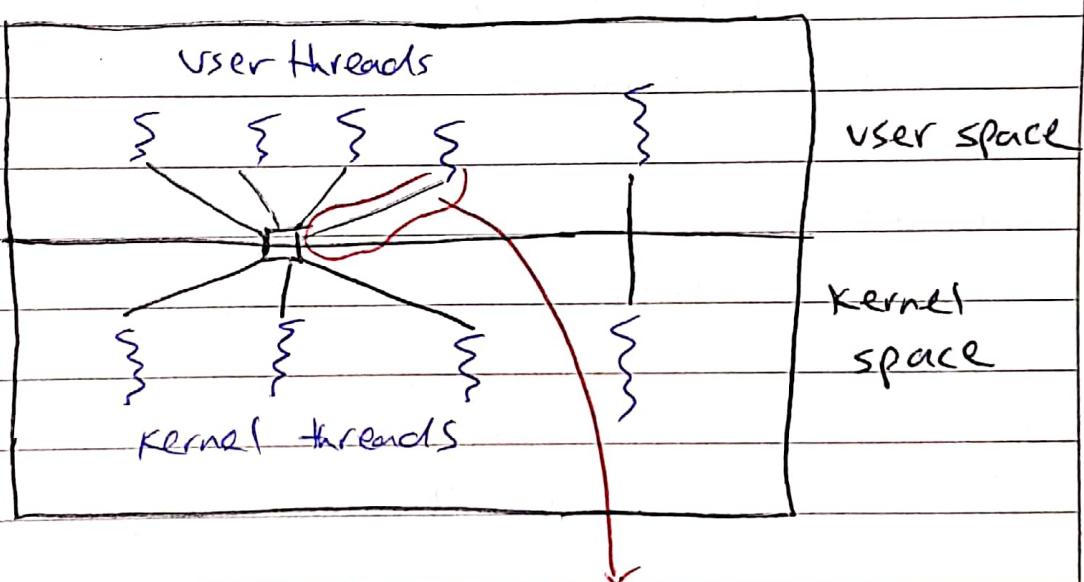
windows with Thread(Fiber) package





one-to-one mapping, many-to-many mapping
one

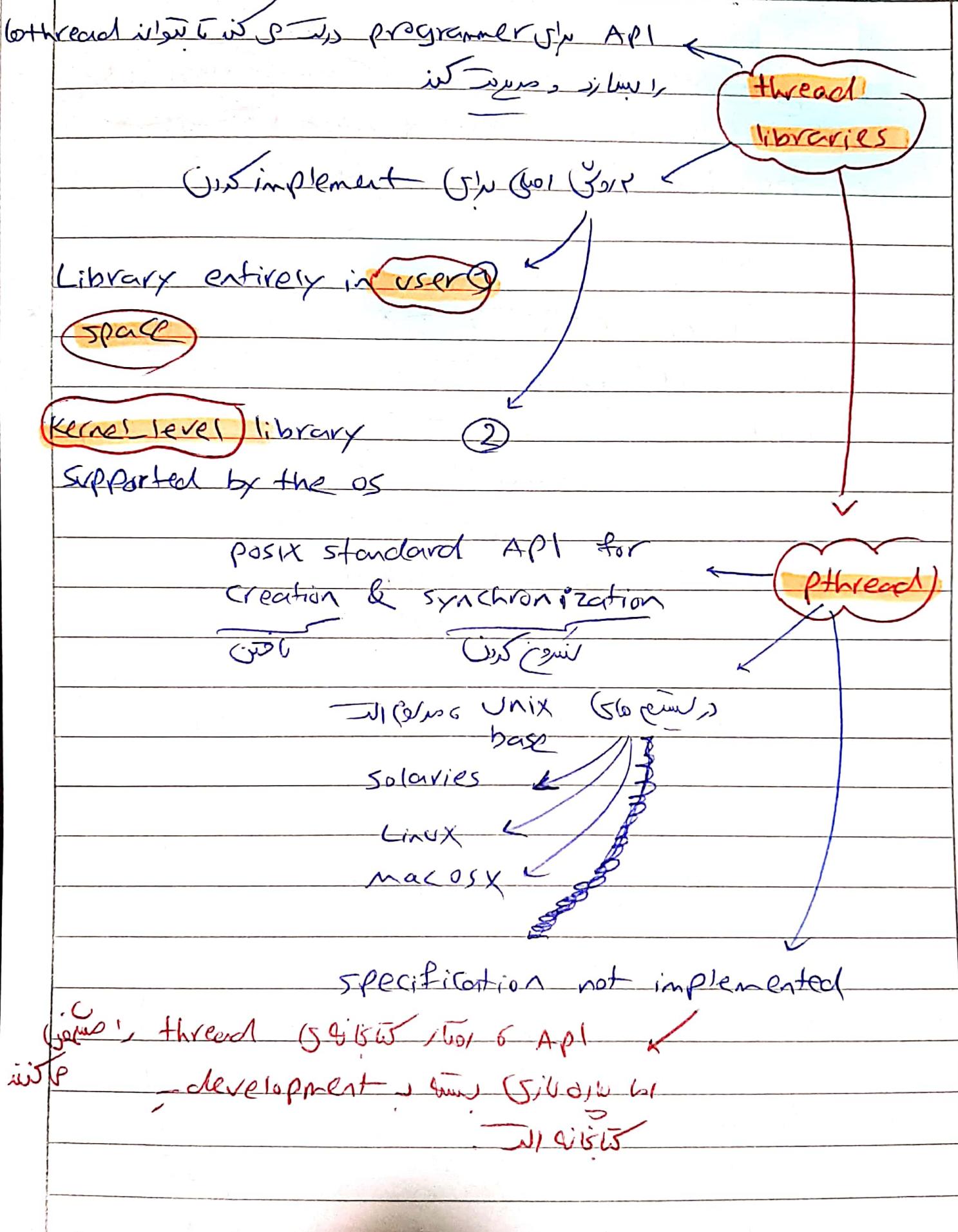
One process (P) has many threads in User space, and one thread in Kernel space. This is labeled "bound".
 One thread in User space can have many threads in Kernel space.
 Every user thread has one kernel thread.

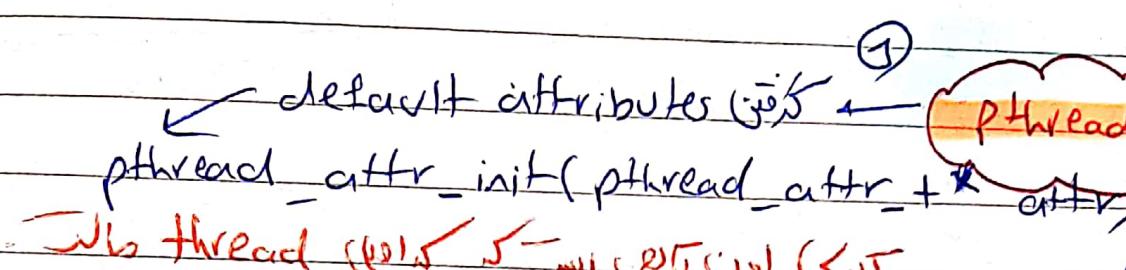


thread is one-to-one thread (j) to b
 One kernel thread is one-to-many
 Many user threads

One user thread is one-to-many
 Many to many

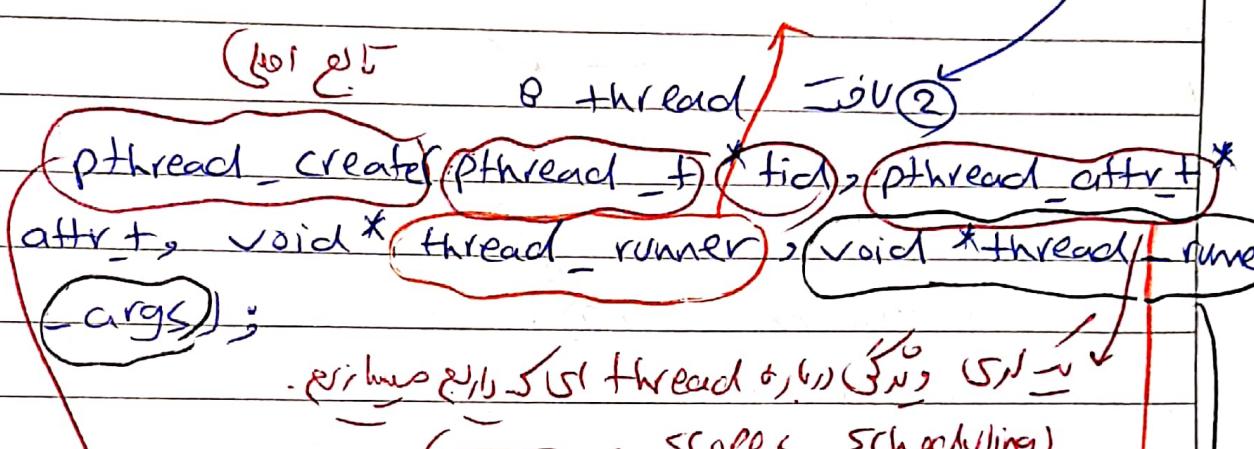
(one-to-many) to (many-to-many)

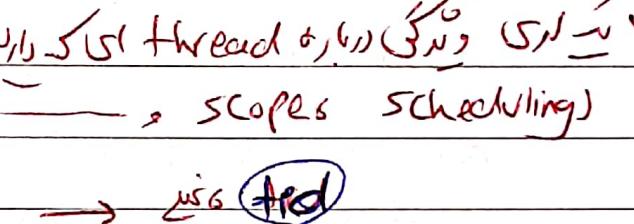


default attributes (جواب ۱) 

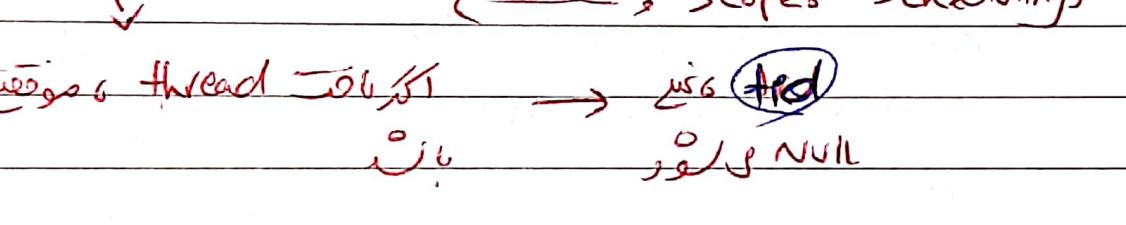
pthread.h

pthread_attr_init(pthread_attr_t *attr)

new thread (جواب ۲) 

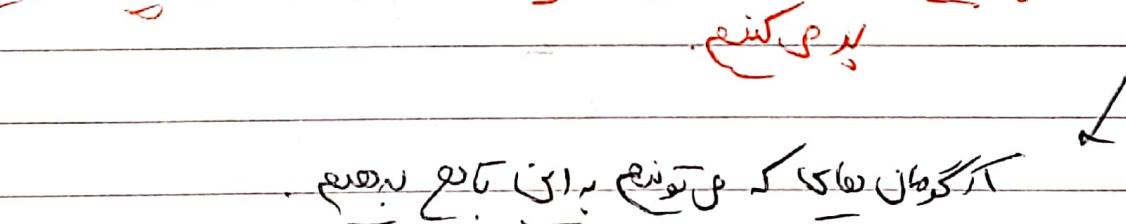
(جواب ۳) 

pthread_join(pthread_t tid, void **val)

new thread (جواب ۱) 

new thread (جواب ۲)

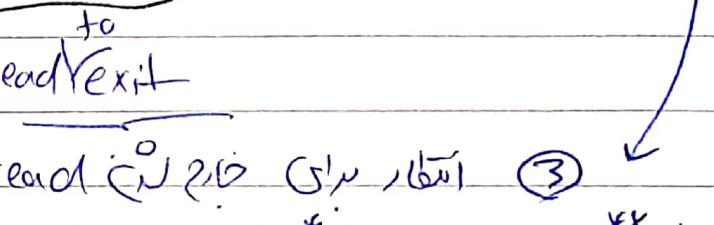
new thread (جواب ۳)

new thread (جواب ۴) 

new thread (جواب ۵)

void * thread_runner

wait for the thread

thread (جواب ۶) 

pthread_join(pthread_t *tid, void **thread_runner_val)

exit thread ④

pthread_exit(void * pthread_runner_ref_val)

thread runner exit (void * runner_ref_val)
→ void * runner_ref_val

(pthread example)

#include <pthread.h>

#include <stdc.h>

int sum; // sum w/ thread
void * runner(void * param); // param w/ thread

int main(int argc, char * argv[]){

pthread_t tid;

pthread_attr_t attr;

threadattr(w)

w type

thread identifier

if (argc != 2)

fprintf(stderr, "Usage %s <integer
value>");

set of
thread
attributes

if (atoi(argv[1]) < 0){

fprintf(stderr, "%d must be >= 0\n",
atoi(argv[1]));

return -1;

}

جواب افریزیت

pthread_attr_init(&attr); (سвойة افریزیت)
pthread_attr_setdetachstate(pthread_attr_t attr, int);

pthread_create(pthread_t *tid, &attr, runner, argv[1]);
pthread_t tid;

pthread_join(tid, NULL); // wait for the thread
process, wait (جایزه) to exit

printf("sum = %d\n", sum);

}

int num ← atoi(argv[1]);

جایزه
= status
جایزه
= status
جایزه
= status

// the thread will begin control in this

void *runner(void *param) { Function
int i; upper = atoi(param);
sum = 0;

جایزه

for (i=1; i<=upper; i++) sum += i; return value

pthread_exit(0);

جایزه

جایزه = status

if (جایزه == 0) thread (جایزه) بسته

ایجاد یک struct که دو فیلد داشته باشد

struct (جایزه) → دو فیلد داشته باشد

first, second

& struct_var

جواب

جواب اسیں کوئی جسے $\text{sum} \leftarrow \text{sum} + \text{arg}$, $\text{pthread_join}()$ تک
تکمیل کرنے والا thread ہے جو اسی میں اور اسی میں

جواب $\text{global} \leftarrow \text{sum}$ میں

example 2

```
#include <pthread.h>
```

```
#include <stdio.h>
```

جواب thread جسے *

```
void * runner(void * args){
```

int arg = *(int *)args; thread کا سینٹ

int j=0; پسکس cast میں اجنبی کوڈ

```
for(int i=0; i<50000; i++)
```

```
printf("%runner %d %d\n", arg, i);
```

```
if(arg==2)  
j=100/j;
```

جسے thread پر کیا?

کسی کو

```
int main(){
```

کوڈ

```
pthread_t t1, t2; thread کی
```

```
pthread_attr_t attr; thread_attr;
```

```
pthread_attr_init(&attr);
```

```
int t1_id=1, t2_id=2;
```

```
pthread_create(&t1, &attr, runner,
```

```
(void *) &t1_id);
```

جسے

```
pthread_create(&t2, &attr, runner,
```

```
(void *) &t2_id);
```

جسے

```
pthread_join(t1, NULL);
```

```
printf("after first join");
```

```

pthread_join(t2, NULL);
printf("End of main");
return 0;

```

و

برازای درکم این احیا thread (سپری شون) روند است ایس نیز پس از این کار چهارم pthread_join()

جستجوی هم احمدی کوئن بدلن thread if it is

لابراوری

کل برخاسته میشود که pthread از thread است کل برخاسته میشود
کل برخاسته میشود جون فناوری آدمی و کارگزاری کل برخاسته میشود
کل برخاسته میشود if(arg==2) . کل برخاسته میشود process
کل برخاسته میشود j=100/j; → برخاسته میشود Runtime error
کل برخاسته میشود کل برخاسته میشود کل برخاسته میشود کل برخاسته میشود

کل برخاسته میشود دو رفع میانجی این اتفاقاتی اندار

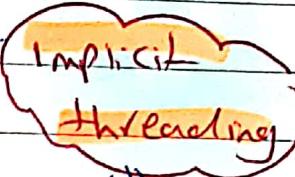
(core dump) floating point exception

(کل برخاسته میشود) (برخاسته میشود)

SESSIONS

وہ thread کو run کر دیتے ہیں جسے thread events کہا جاتا ہے اور اسے کہا جاتا ہے

وہ thread کو run کر دیتے ہیں جسے thread events کہا جاتا ہے اور اسے کہا جاتا ہے



وہ Compiler وہ thread events کو run کر دیتے ہیں اور اسے کہا جاتا ہے

run-time libraries

method یا فنکشن

thread pools

OpenMP

Fork-Join

ان بھت سے رہائشی مکان

ملکوں

X Grand Central Dispatch

thread ہمارے سامنے آتے ہیں

Intel threading Building

کام کرنے والے ہیں

X blocks

thread ہمارے سامنے آتے ہیں

دیگر

جسیں ہمارے سامنے آتے ہیں

کوئی ہمارے سامنے آتے ہیں

core کو کام کرنے والے ہیں

ایک thread کو کام کرنے والے ہیں

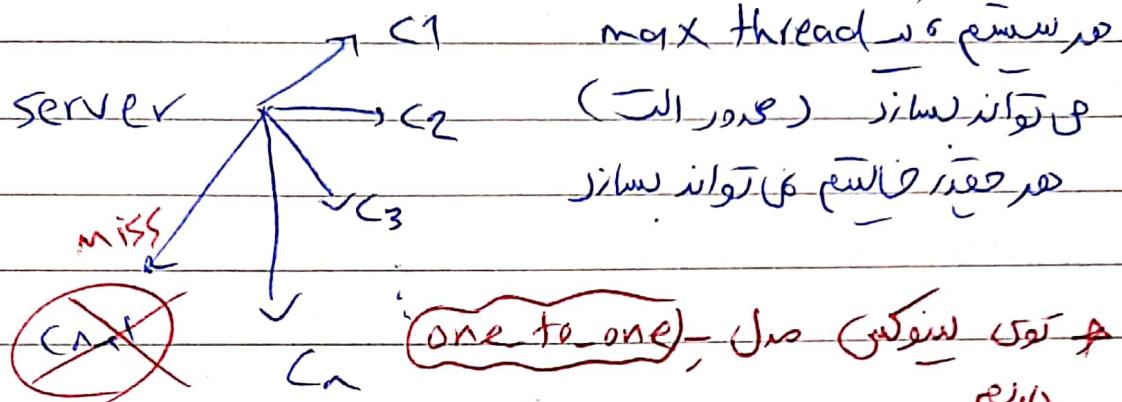
ایک thread کو کام کرنے والے ہیں

کوئی ہمارے سامنے آتے ہیں

hirmandpaper

Client-server example

نحوه داد و تعداد سریع رایج است این مدل برای داد و سریع داد و سریع است.



این یعنی C_n کی لمحہ کرنے کا دورانیہ thread خاصی کی تعلیم

سماں اعلیٰ \rightarrow اگر صدای کوئی نہیں کرے تو ڈھونڈنے کا طریقہ
والفیں در حال اجرا ہونے والے تصور (آئندہ پاسخ) کو نہیں کرے
وہی client ای کوئی thread جب خواہ دنار کو اپنے

ایک جیسے thread \rightarrow pthread_create() سے خواہیں کرے کہ thread کی کامیابی کا مطابق (ٹائین کوئی) سے ہے

(ایک خطا کی زمانہ run-time)



اگر صورت سنی \rightarrow اعمال کلائیٹ کی ایسیں پر اور

اگر سوانح کوئی کوئی کیم کرے تو ایسی خطا کی حالت

یعنی پس پیش کیم خطا کی درستی لئے کیم از خارج ایسا

گروپ thread توں pthread_create() کی پڑھ کر جو،
ساز سکنے کا اور جو while loop کی طرف
تک ازدرا رہا thread_id کے طبق time ہو thread
کی سرعت کا اندازہ کرو و بعد بتوانے
کیلئے join کرے جو thread
این سے راہ حل دستی اسے

حالہ تابی انسٹرکشن کا بجا ہوئے (سی) ای نہ رکھ رہا صورت کیم ←
جس کی وجہ سے اس کے لئے کم کرنے کا رکنہ
میں توں نہیں کیا رکنہ library

فہمی

implicitly

لے thread id اس ستر (اسے) میں ہو جائے

Thread pools

معنی کے طبق thread (سکریپٹ) فرما رہا تھا کہ اس کے ساتھ ہے

گروپ assign کے thread ہے task کو کہا جائے

task کیم کیا کہ اس کے لئے thread ہے

کہ اس کے لئے task ہے جس کو کہا جائے

کہ اس کے لئے

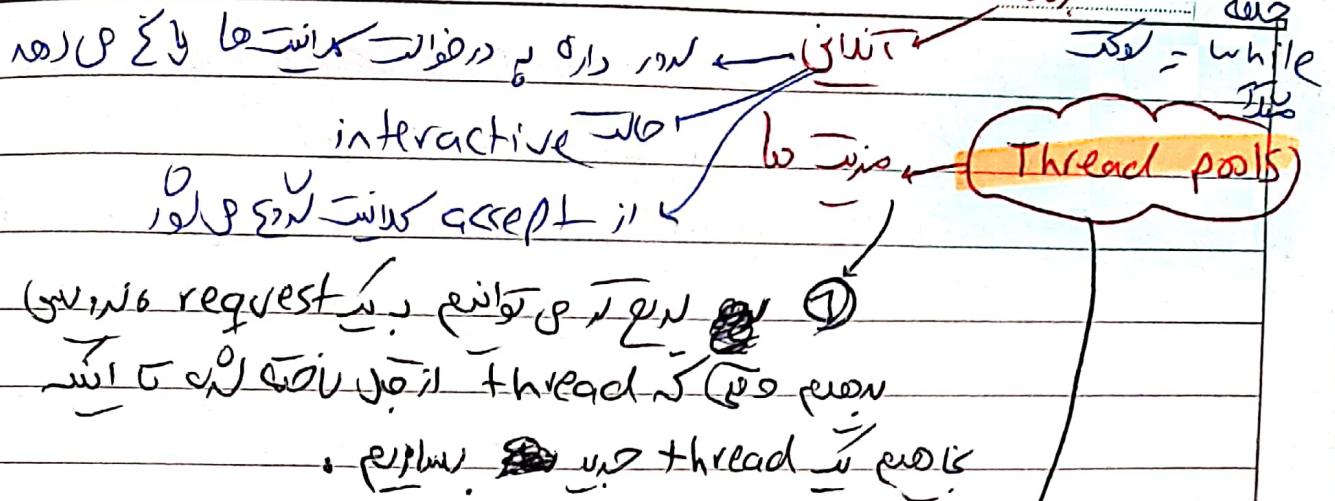
Handler کی گفتگو کی کوئی مالکیت نہیں کی

کہ data structure کی کوئی مالکیت نہیں کی

کہ اس کے لئے task ہے اس کے لئے thread ہے

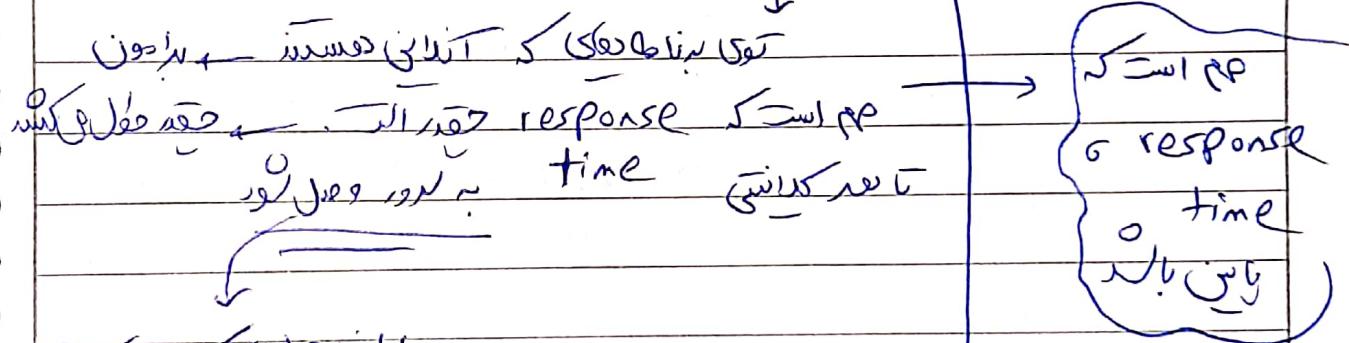
کہ اس کے لئے task ہے اس کے لئے thread ہے

لورکسٹ - accept \rightarrow قبول از \rightarrow آفیلن (available) \rightarrow بینگن (binding) \rightarrow قبل از \rightarrow قبول از



عمر تکمیل در زمان آفلاین این thread pool را ایجاد کنیم

وچی هنوز وچی هنوز.

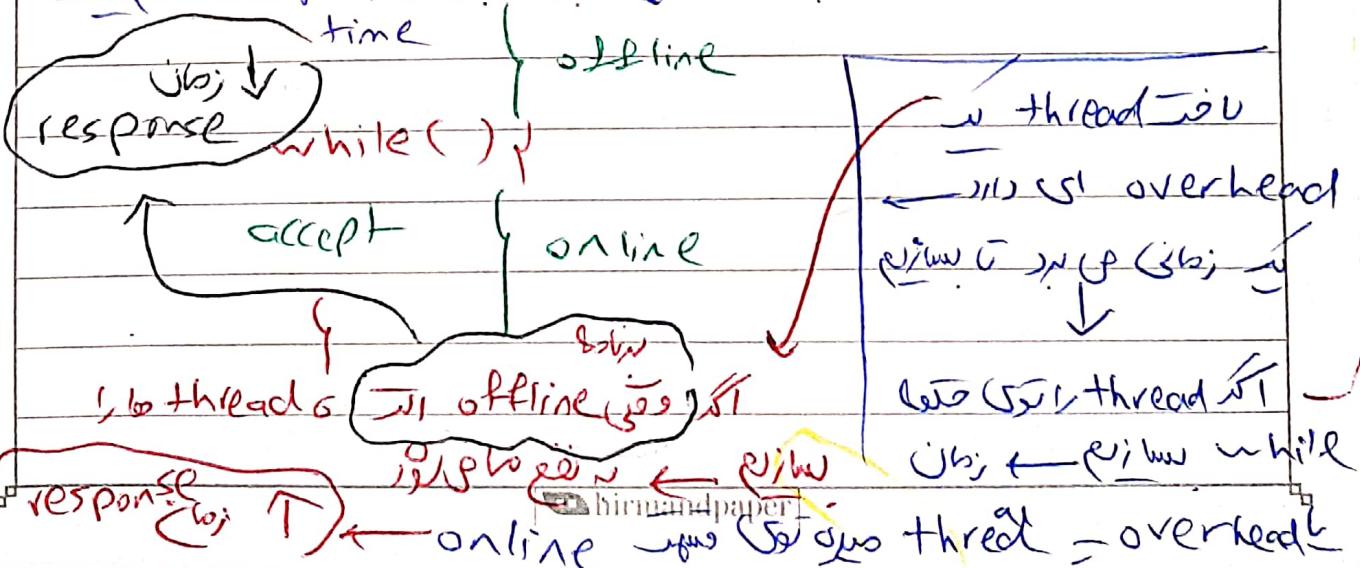


آفلاین جابجایی کردن در برای کلاس

کم خواهد بود.

(خطای) که بنده آفلاین است \rightarrow در حقیر، دوام فعل داشتیں

اصدیقی داشتیں \rightarrow یعنی **response** (رد پاسخ) \rightarrow کم خواهد بود.



جواب → پریولیٹری thread کو (کوئی کام نہیں) کام کرنے والا task ہے

to task کو کہا جاتا ہے

(global)

thread pool

task pool کا size (50) ہے

کام کرنے والے task کو task pool کو کہا جاتا ہے

- max کو کہا جاتا ہے (کام کرنے والے task کا limit)

کام کرنے والے task کا limit

کام کرنے والے task کو assign کرنے والے thread کو task

کام کرنے والے task کو task

task کو assign کرنے والے thread کو task

کام کرنے والے thread کو task

task کو assign کرنے والے thread کو task

کام کرنے والے thread کو task

task pool کو windows API

Concurrency

task

Sample thread pool API

create thread

void first_task() { task1 }

void second_task() { task2 }

void third_task() { task3 }

main()

max

pool + p(2);

thread

add

task

pool

bind

task

thread

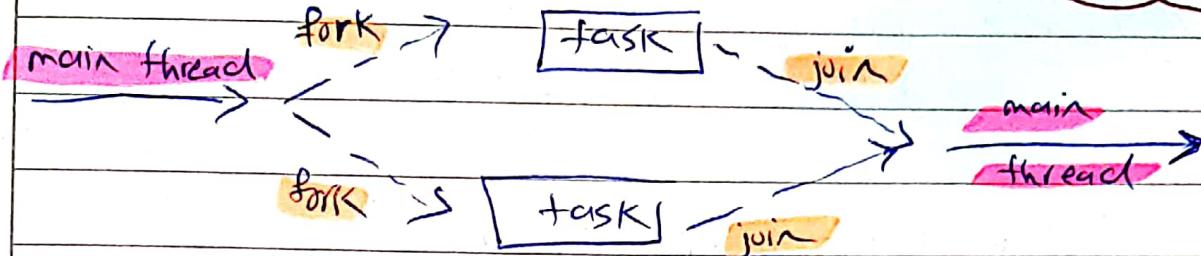
pool

hirmandpaper

third

is fork a (task u) thread (y/n)
is join part of your

Fork-join
parallelism



Slow task → new problem → subproblem → subproblem → ... → subproblem → join → result (final result)

general algorithm for fork-join

Task(problem)

if problem is small

solve the problem directly

else

subtask1 = fork(new Task(subset of problem))

subtask2 = fork(~ ~ ~)

result1 = join(subtask1)

~ 2 = join(subtask2)

return combined results.

merge sort,
quicksort

is this a (task u) thread (y/n), (is task u doable)

is it a subtask → no

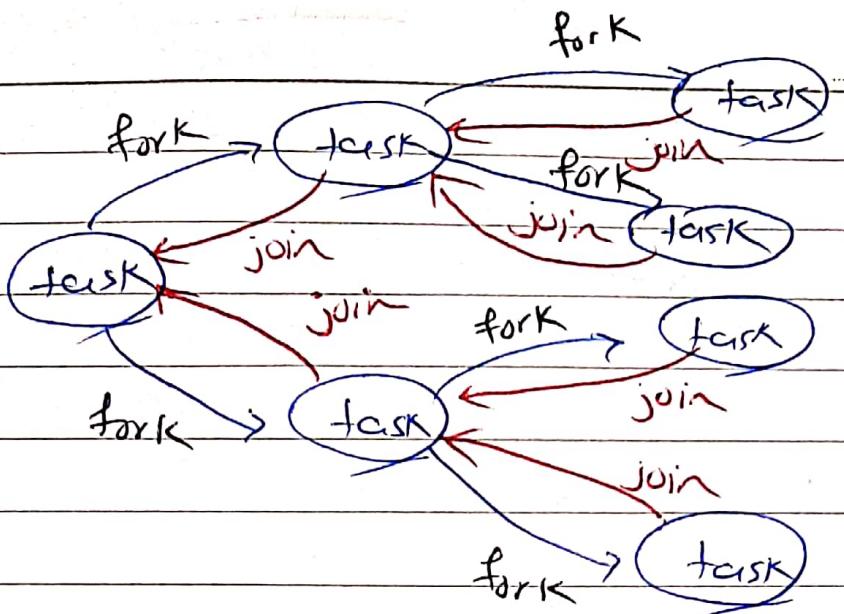
divide & conquer

fork → new problem (doable) → no

new task (doable)

handpaper

new task (doable)



public SumTask() { fork-join equation
using java API
y
j

sumTask leftTask = new sumTask(beginmid,
array)

sumTask rightTask = new sumTask(midf1, end,
array)

leftTask.fork();

rightTask.fork();

return rightTask.join() + leftTask.join()

data parallelism ← data parallelism, with multiple cores

OpenMP

C/C++ API, compiler directive; (JRW)

→ loop, parallel programming

→ parallel block (go) ← parallel block (JRW), parallel regions

→ new (JRW), (gilgo)

#pragma omp parallel

Create as many threads as there are cores

hirmandpaper

directive

pragma op parallel for

```
for(i=0; i<N; i++){
    c[i] = a[i] + b[i];
}
```

run for loop in parallel

میں کہوں گا
اچھا ہے جیسا
کہ ہے

threading issues

کوئی بھی کوئی fork کی کسی نہ
exec کوئی بھی کوئی multithreading

بروز روند ← Signal handling
کوئی بھی ← Context

Asynchronous ← thread cancellation of
deferred ← target thread

Crossing fence of std API calls

↑
semantic of fork

1. fork کی thread کو fork کی

exec

2. both threads are duplicates doing

nil fork کی وہی کام کرے جو اسی کام کرے

duplicate thread کو thread کو ① کرے

کو thread کو کام کرے ②

done invoke & fork()

in thread میں exec

exec()

جنہی براہی حال اجرا کرنا میں ہے exec
جسیں کہ بروہی دس

وہی fork اور exec کا مضمون فلسفی ہے ،

وہی thread کا fork اور exec

Kill -1 دھای سے ریکارڈ

shell میں ہے ہماری signal

Ctrl + C

انہیں ہماری signal

signals handling

event کرنے کے لیے notify , process

مسنون انفاس افسارہ

بردازی و یا کردار

از پڑی بڑی بروہی کرنے کے لیے signal handler

event سے signal

process سے signal

signal handler کے

default (1)

User-defined (2)

کامیابی کے لیے signal

کامیابی کے لیے signal

kernel کے default handler

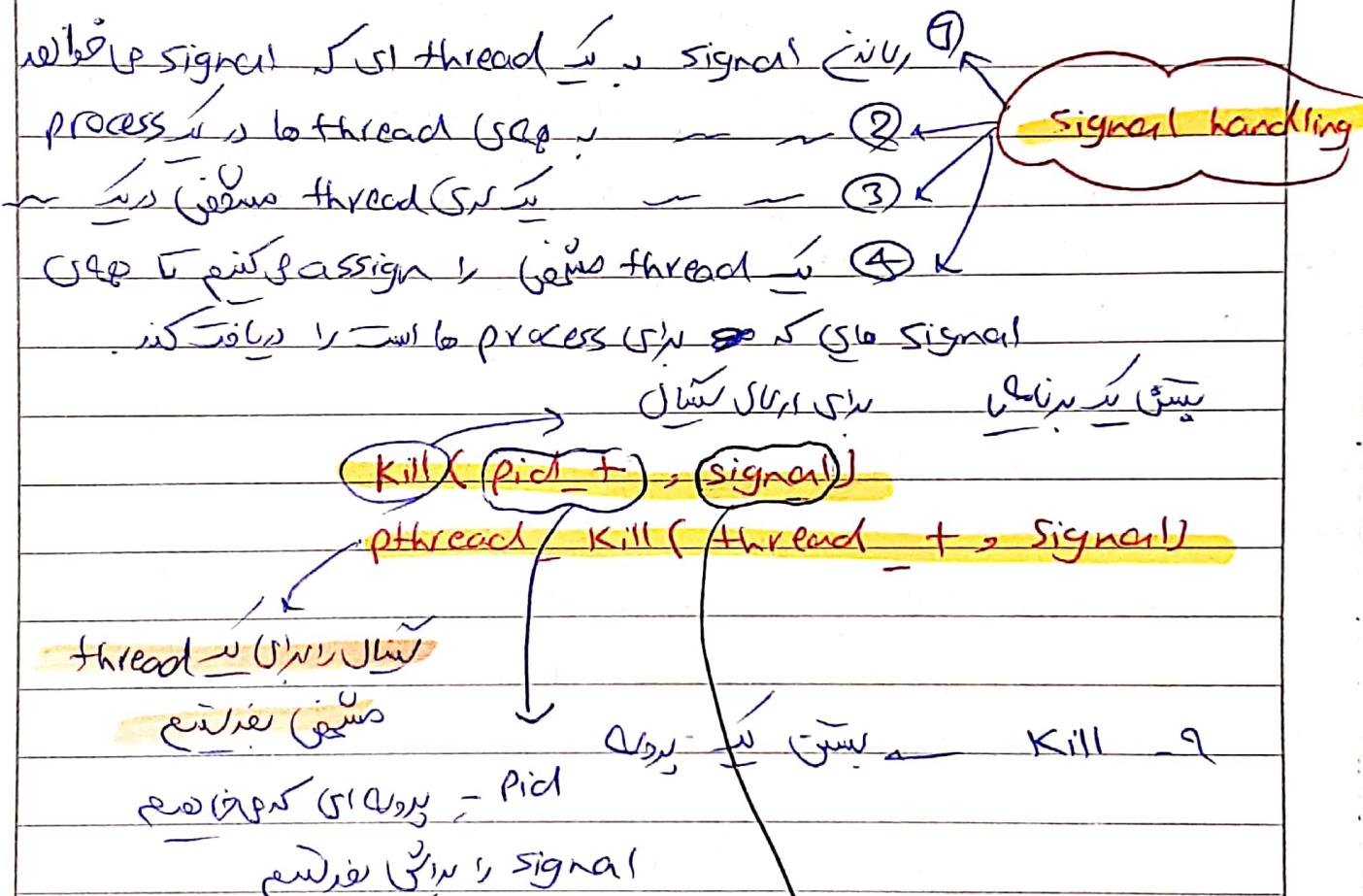
کامیابی کے لیے signal

default handler کے تعلق ہوئے signal handlers

in override

process \rightarrow a signal \leftarrow single-threaded (S) \times
signal delivered

threading (S) \rightarrow process \leftarrow signal \leftarrow thread (T) \leftarrow signal
one thread can have multiple threads \leftarrow thread (T) \leftarrow signal
multiple threads can have one process \leftarrow process (P) \leftarrow signal



signal \rightarrow S \rightarrow signal handler \rightarrow action
either (S) - signal

Java Cancel Thread ← خریداری thread از وال

کل

نیو Thread کیا کرنا thread'ın terminate

کیا thread'ın فری کرنا Thread دھال کرنا ہے ①

کیا thread'ın اسے record کرنا ہے ②

کیا

thread

cancellation

کیا thread'ın کو web page ہے ③

کیا thread'ın کو browser کے record میں دھال کرنا ہے ④

کیا Cancel کیا کیا thread کو cancel کرنا ہے ⑤

کیا cancel کیا کیا thread کو cancel کرنا ہے ⑥

target thread

cancel cancellation

Asynchronous cancell ① ← cancel cancellation

Terminates the target thread
immediately

کیا target
thread

Deferred cancellation ②

target thread

کیا target thread کیا کیا timer

کیا target thread کیا کیا signal cancellation

Exception interrupt handling ← Signal

signals cancel signals, thread cancellation will invoke handler
state (cancel) thread is (cancel)

in (cancel) thread

Mode	State	Type
off	Disabled	-
Deferred	Enabled	Deferred
Asynchronous	-	Asynchronous

→ disable → cancel thread
→ thread waiting (pending) now cancel will be enabled

default type

thread is ready (for cancellation)

cancellation point

EX: pThread testcancel()

EX: read function

Cancellation info

if (cancel) exit

if thread cancellation (cancel) signal

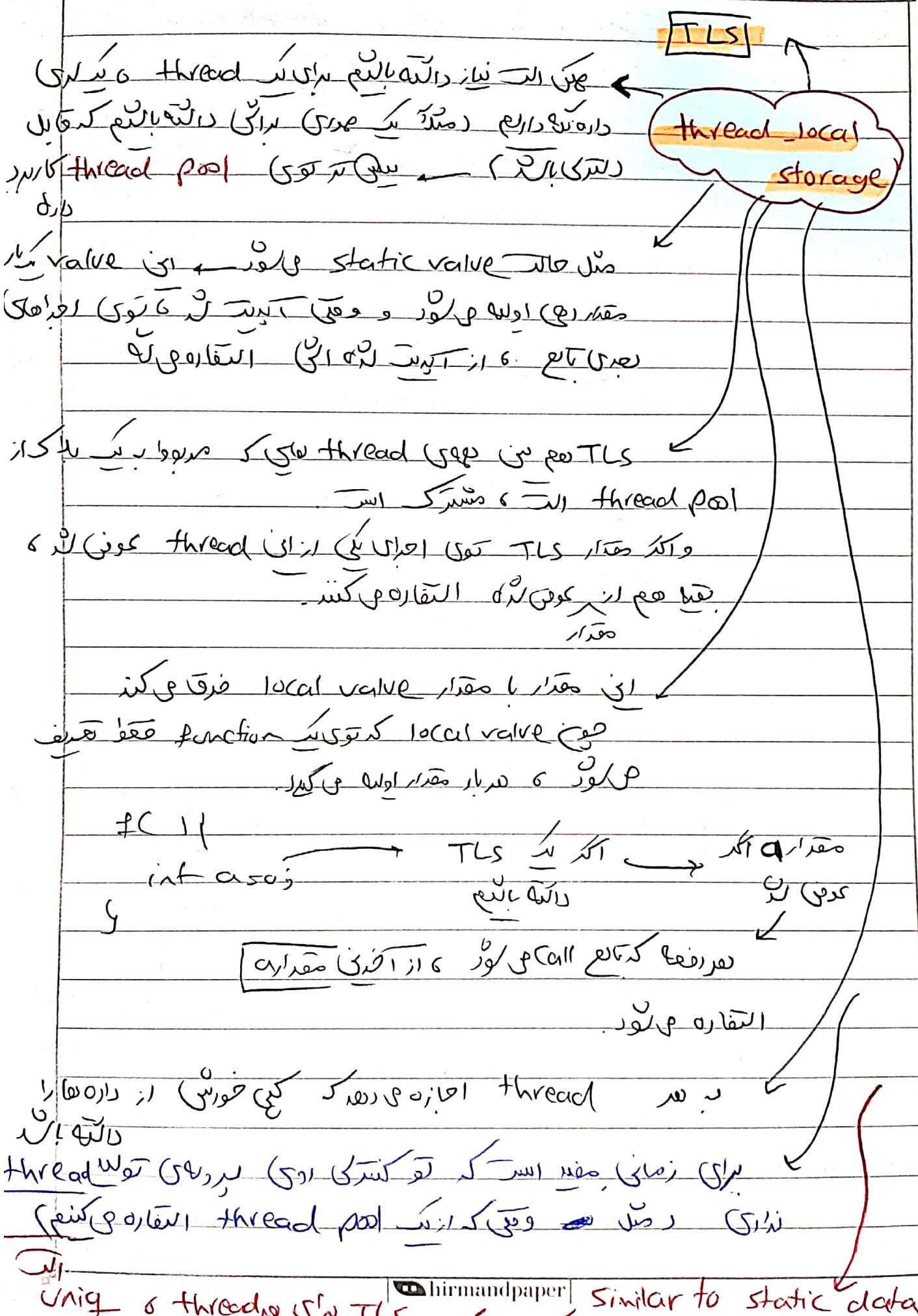
if (cancel) signal

stop read

if read bin (input) std::set<char>

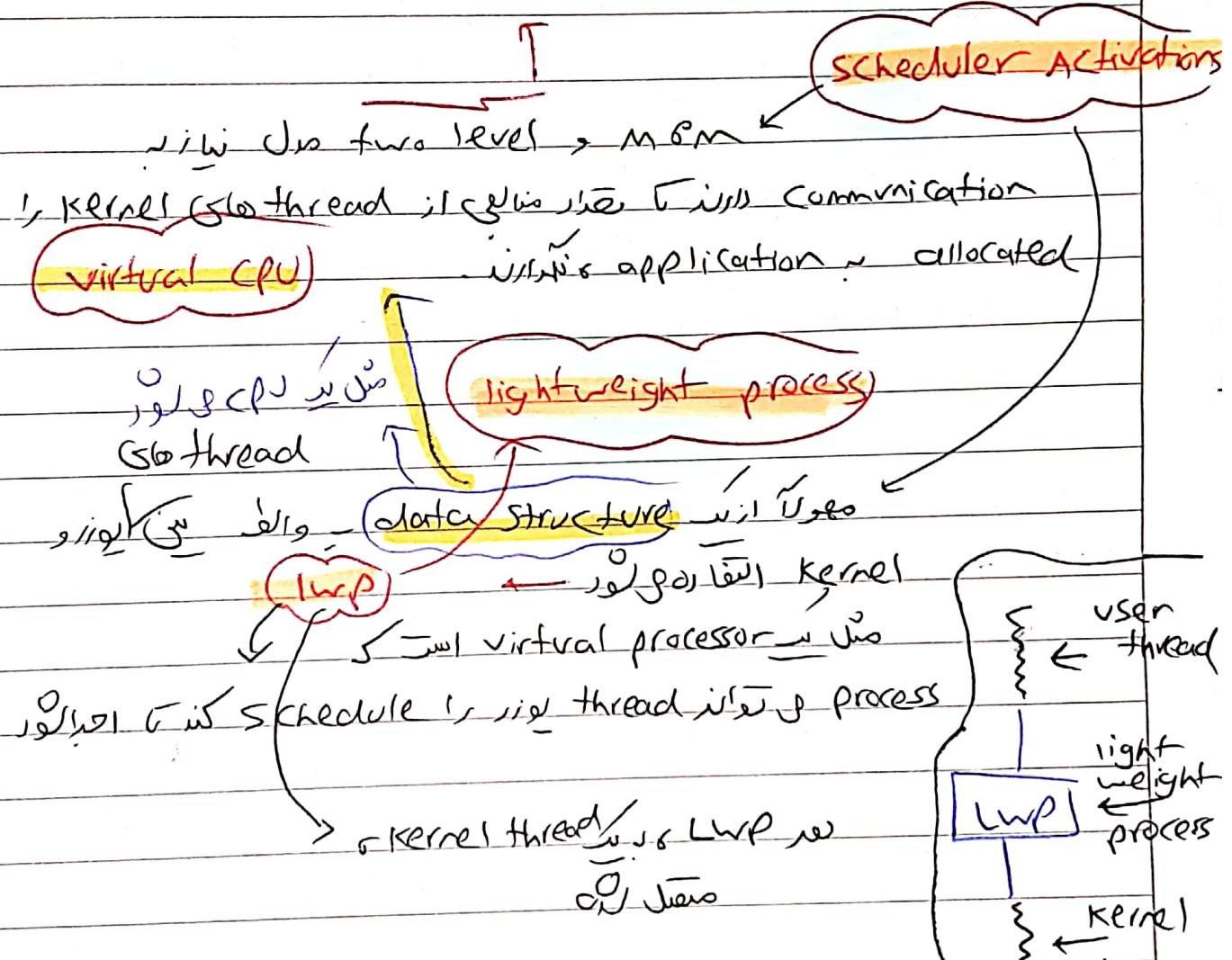
if char is in set then remove it from set

if char is not in set then add it to set



6. local variable invocation is visible to local variable
 6. local variable invocation is not visible to local variables

Kernel thread <--> User thread one-to-one



user thread is mapped to virtual processor which is mapped to kernel thread

user multithreaded multitasking is mapped to kernel thread

kernel thread is mapped to task

task is mapped to core

کیمیا میں لفظ "User thread" کا معنی "user library" کے لئے کہا جاتا ہے۔

user thread کو اختراع کہا جاتا ہے۔

CPU intensive tasks کو CPU intensive tasks کے لئے کہا جاتا ہے۔

user thread کو CPU intensive tasks کے لئے کہا جاتا ہے۔

user thread کو CPU intensive tasks کے لئے کہا جاتا ہے۔

user thread کو CPU intensive tasks کے لئے کہا جاتا ہے۔

user thread کو CPU intensive tasks کے لئے کہا جاتا ہے۔

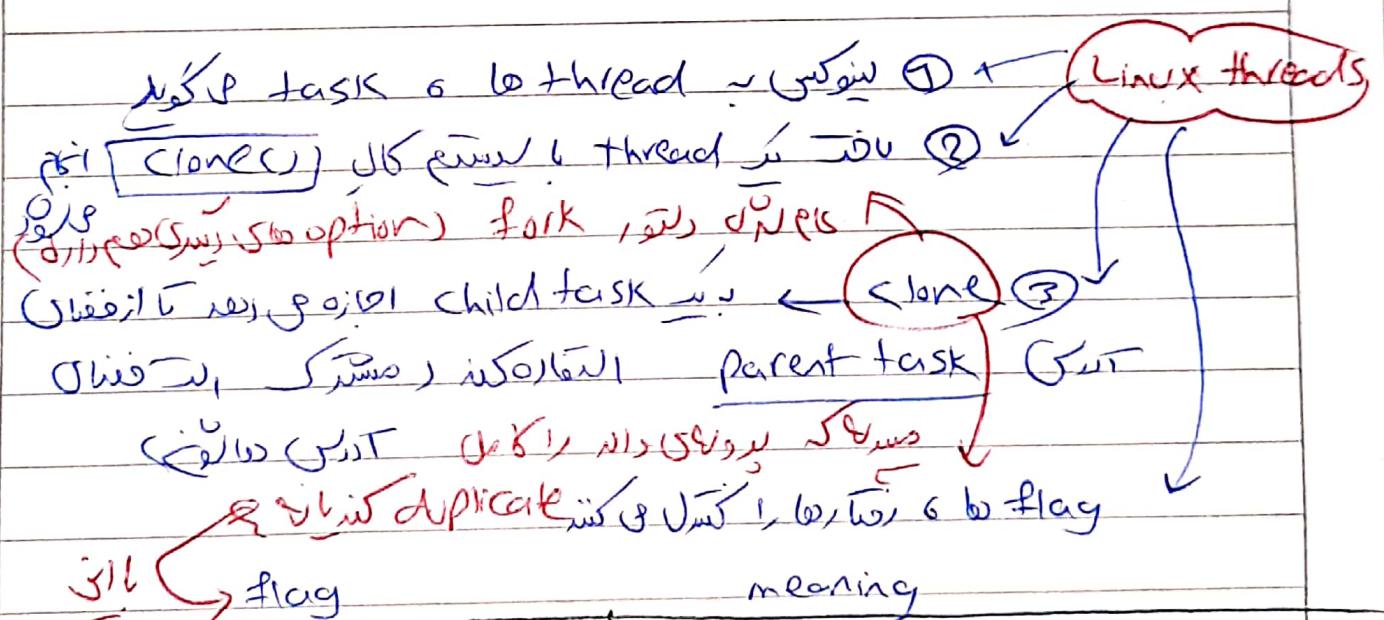
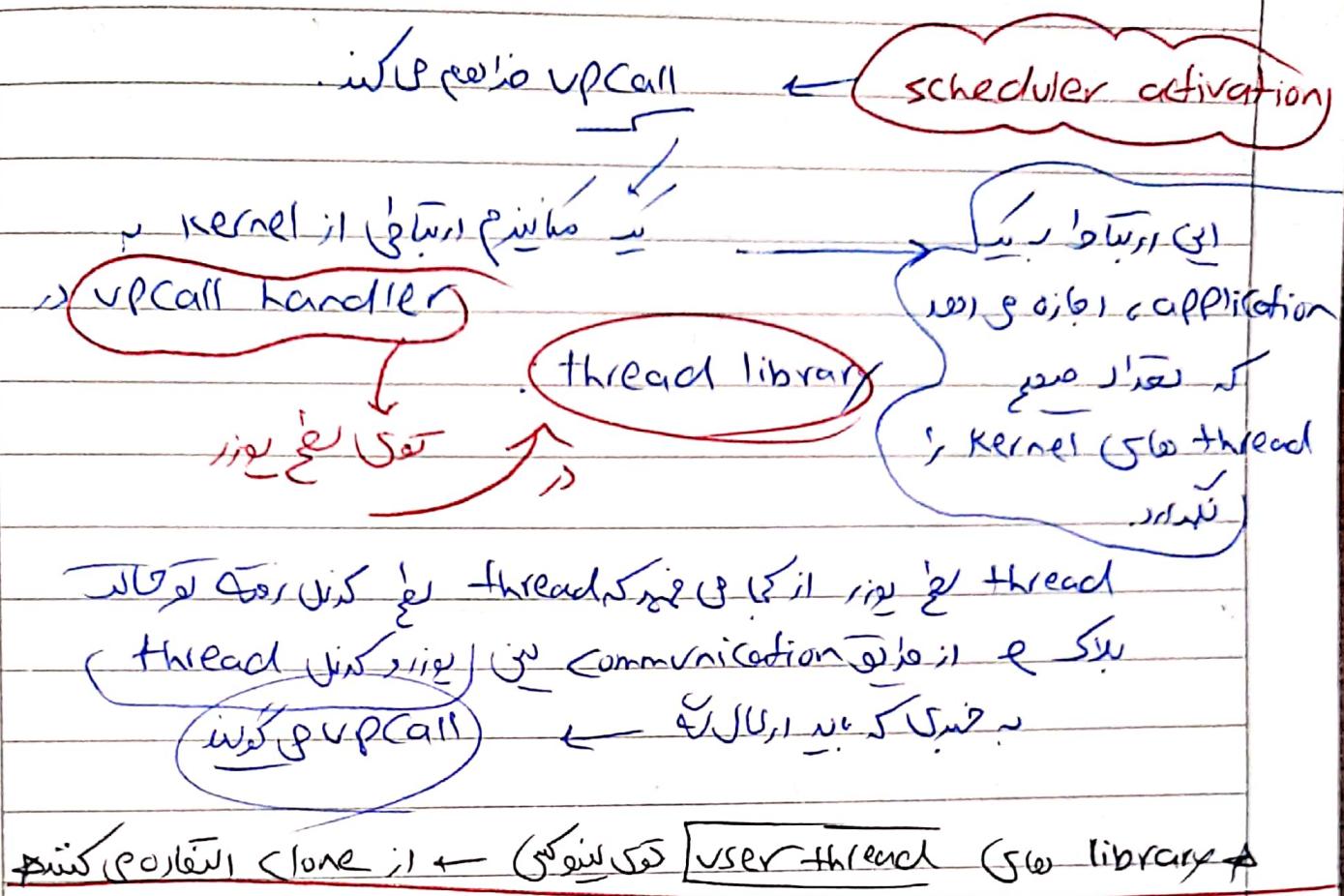
user thread کو CPU intensive tasks کے لئے کہا جاتا ہے۔

user thread کو CPU intensive tasks کے لئے کہا جاتا ہے۔

user thread کو CPU intensive tasks کے لئے کہا جاتا ہے۔

user thread کو CPU intensive tasks کے لئے کہا جاتا ہے۔

user thread کو CPU intensive tasks کے لئے کہا جاتا ہے۔

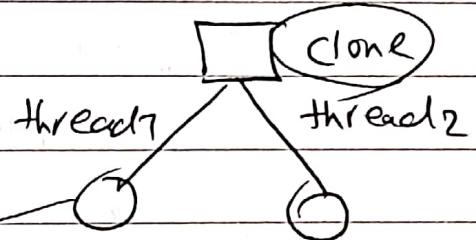


(w/ flag)	clone_FS	file system information is shared
	CLONE_VM	the same memory space is ~
	CLONE_SIGHAND	signal handlers are shared
	CLONE_FILES	the set of open files is shared

struct task_struct → points to process data structures. (shared or unique)

↳ pthread (سیسی ایکس سی) ↪
getconf GNU-LIBPTHREAD-VERSION

↳ thread (سیسی ایکس سی)



When running, pick 1 thread at a time

Follow (فولو) ↪ next (سیسی ایکس سی)

(to pick)

(من کو ہو تو thread یو context switch)

(بھالے یو context switch تھا جس پر)

قابل clone یو گزینے کو container, virtualization

+ Linux

SESSION 10

< scheduling >

void *runner(void *args)

int arg = *(int *)args;

int sum = 0;

for (int i=0; i<10000000; i++) {

 دیکھو اس کو کیا کرے

 sum += i;

 // printf("sum %d\n", sum);

g

g

8.3 چند thread یکت

```
int main() {
    int thread_num = 50;
    pthread_t *threads;
    pthread_attr_t thread_attr;
    pthread_attr_init(&thread_attr);
    for (int i=0; i<thread_num; i++) {
        pthread_create(&threads[i], &thread_attr,
                      runner, (void*) &i);
    }
    for (int i=0; i<thread_num; i++)
        pthread_join(threads[i], NULL);
    return 0;
}
```

رجوع (جیف) CPU ← این را کنم اگر نه
Usage

جیف نیز چنگ، چند thread (جیف) را در یک CPU کار کنید
و چند استان (جیف) CPU ← این را کنم (جیف) تولید
کنید (جیف) چند core را در یک CPU کار کنید
و

6CPU یک Core را بخواهیم، ۱۰۰% ← میتوانیم TOP

% CPU 0 49.5

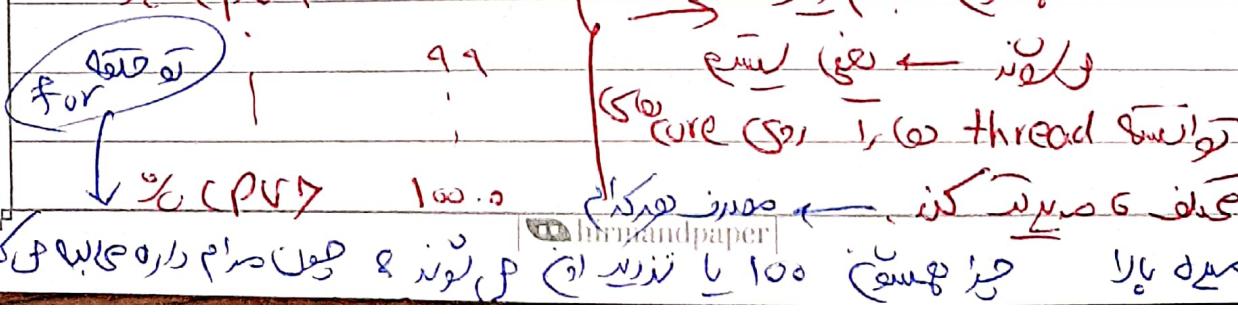
% CPU 1 100.0

% CPU 2 99

% CPU 3 99

% CPU 4 100.0

% CPU 5 100.0



لما حصلت على مساحة مalloc تردد المبرمج في إدخال المحتوى المطلوب في المكان المخصص له

لذلك ينصح بـ `malloc` و `free` و `realloc` و `calloc` من حيث المدخرات

لذلك ينصح بـ `malloc` و `free` و `realloc` و `calloc` من حيث المدخرات

لذلك ينصح بـ `malloc` و `free` و `realloc` و `calloc` من حيث المدخرات

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

الآن نلقي بالنظر إلى مساحة الـ `cpu_usage` التي تم إدخالها

و CPU باری با IO bound که سریع است و CPU باری با CPU intensive که سریع نیست.

و CPU باری با IO bound دارند و نیز CPU bound دارند.

CPU usage \rightarrow IO bound که سریع است
(این خود را بین چندین CPU بیندازد)

و switch task کرد که سریع است

و switch task کرد که سریع است
که سریع است و قابل برداشت است

و switch task کرد که سریع است
که سریع است و قابل برداشت است

IO bound

و

سریع است و قابل برداشت است

CPU bound

سریع است و قابل برداشت است

و

سریع است و قابل برداشت است

load store

add store

read from file

wait for IO

store increment

index

write to file

P₀

P₁

CPU burst

I/O burst

CPU
burst
P₁

CPU burst

wait for IO

I/O burst

load store

add store

read from file

CPU burst

wait for I/O

I/O burst

الخطوة الأولى هي إدخال جميع المدخلات إلى CPU

الخطوة الثانية هي إخراج جميع الناتج من CPU

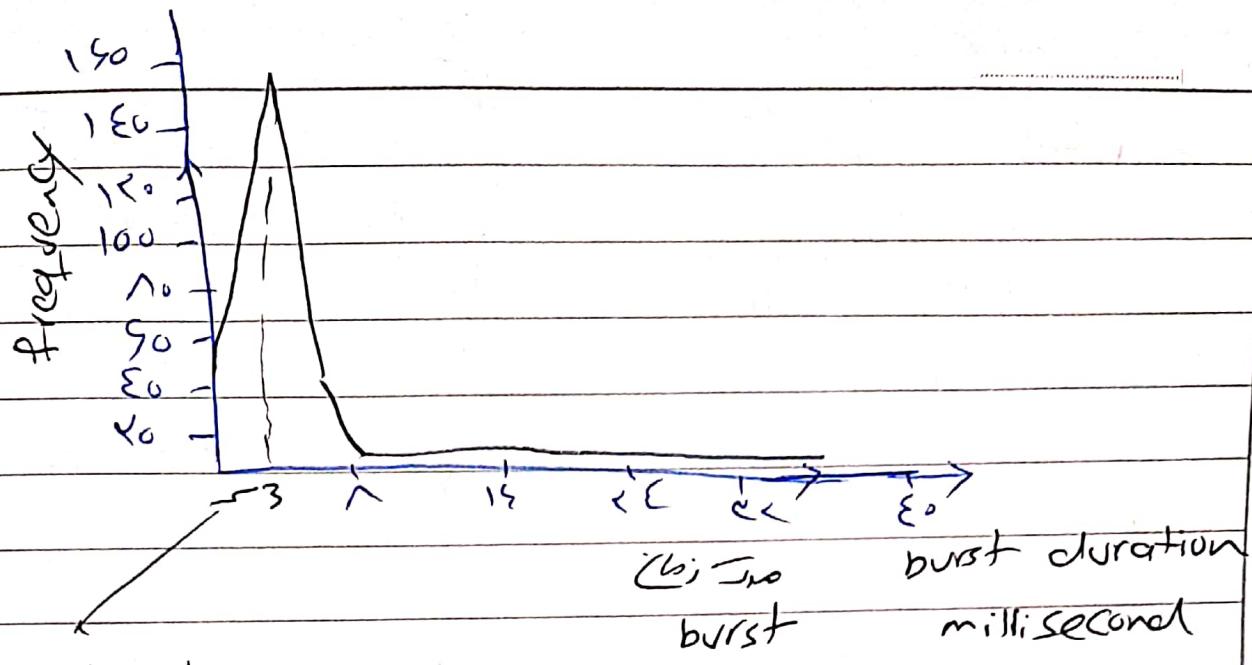
(1) I/O burst P₀ → CPU burst P₁

الخطوة الثالثة هي إدخال جميع الناتج من CPU إلى P₁

الخطوة الرابعة هي إخراج جميع المدخلات إلى CPU

CPU burst distribution

CPU



بوجburst
فرکانسی (frankness) \rightarrow ms

بوج \rightarrow باری طبعی (natural burst)
 \downarrow
(frequency) \downarrow

این عدد نشان می‌دهد که اکثر burst در بین ۰ تا ۱۶۰ ms
کوتاه مدت (short) و فرکانسی (frequency) های که burst مولای صردارند، خیلی کم است.

I/O داری \rightarrow burst CPU

از این دلایل سیستم واقعی قدرتمندی برای scheduling

التفاوت کنید

برای کامپیوچر \rightarrow CPU = burst

و سایر دستگاهات I/O

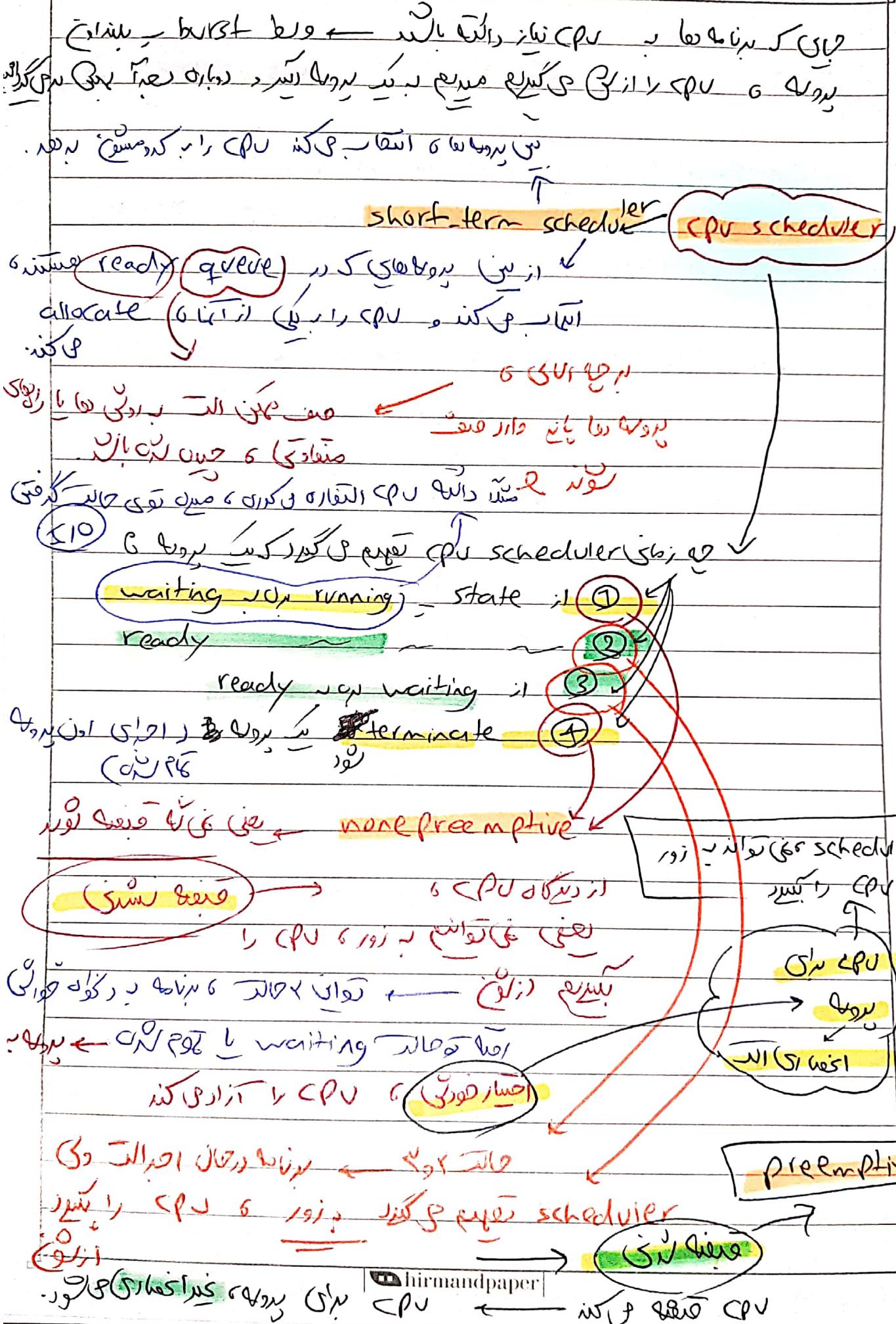
برای CPU - burst اگر بعد از ۰.۱ sec و این بعد از ۰.۲ sec ...
برای CPU - burst اگر بعد از ۰.۳ sec ...

اما در scheduling \rightarrow این مورد را نمی‌دانیم

برای CPU - burst \rightarrow بین ۰.۱ sec و ۰.۲ sec

برای CPU - burst \rightarrow بین ۰.۱ sec و ۰.۲ sec

برای CPU - burst \rightarrow بین ۰.۱ sec و ۰.۲ sec



اگر از Preemptive scheduler است \rightarrow چنین الی می شود

بروک اول در حال اجراهه از سرور CPU \rightarrow اگر از پر سرور CPU بسیار نارمل تر عرض کند \rightarrow بروک دوم نیز شروع کند و بروک اول بعد از آن شروع کند \rightarrow این نتیجه را می خواهیم کرد

در Kernel mode برای preemption اگر \rightarrow بروک اول در درون کرنل اجراهه و بروک دوم کرنل های کرنل

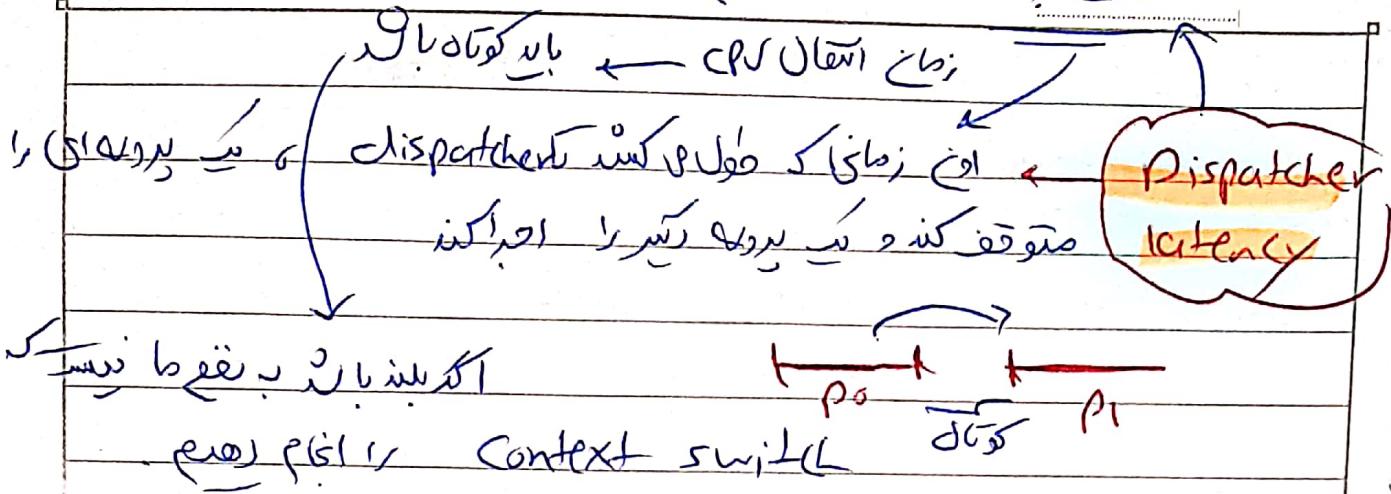
بروک اول در data structure بروک دوم در data structure \rightarrow بروک اول در درون کرنل اجراهه و بروک دوم در خارج از کرنل \rightarrow بروک دوم اینجا اجراهه شود

حتی اگر OS می خواهد این را کند \rightarrow این ممکن نیست

Short-term-scheduler \rightarrow Concurrency در user mode \rightarrow Dispatcher \rightarrow CPU scheduler \rightarrow Dispatcher \rightarrow user mode

۳ jumping to proper location in the user context switch \rightarrow ۱ program to restart switching to user mode the program \rightarrow ۲

نیپر Dispatcher



وہی سیستم افلاک اور سیکنڈز میں سیکنڈز میں افلاک اور سیکنڈز میں افلاک

virtual memory → vmstat 1 3 ①

status cat /proc/pid/status ②

CPU usage

وہی سیستم افلاک اور سیکنڈز میں افلاک اور سیکنڈز میں افلاک

Context switch

وہی سیستم افلاک اور سیکنڈز میں افلاک اور سیکنڈز میں افلاک

وہی سیستم افلاک اور سیکنڈز میں افلاک اور سیکنڈز میں افلاک

وہی سیستم افلاک اور سیکنڈز میں افلاک اور سیکنڈز میں افلاک

وہی سیستم افلاک اور سیکنڈز میں افلاک اور سیکنڈز میں افلاک

vmstat → Context switching 6 سیکنڈز میں افلاک اور سیکنڈز میں افلاک

Context switch ← CS ← boot

وہی سیستم افلاک اور سیکنڈز میں افلاک اور سیکنڈز میں افلاک

وہی سیستم افلاک اور سیکنڈز میں افلاک اور سیکنڈز میں افلاک

وہی سیستم افلاک اور سیکنڈز میں افلاک اور سیکنڈز میں افلاک

vmstat ① ② → دھل دھل و درکل

وہی سیستم افلاک اور سیکنڈز میں افلاک اور سیکنڈز میں افلاک

وہی سیستم افلاک اور سیکنڈز میں افلاک اور سیکنڈز میں افلاک

CS ← Thread

کے احتراں فوریوں بوجہ context switch

voluntary - ctx - switches & 20305
non voluntary

بزیر انجام نہ کرے context switch
Scheduler

Scheduling criteria

(وزیر صرف برداشت) CPU utilization

busy CPU (جیسا کہ ماتحتیں) کے لئے

burst (باڑھ) throughput

نکار برداشتی کے احتمالی را در واحد زمان کا
کامل فروخت

نکار برداشتی کے احتمالی زمان (turnaround time)

CPU waiting time

مقادیری کے سبب (مخفوق) مولی کسہ بالآخر اور
زمانی کے مولی کسہ تاں دوڑ کا احتمال

waiting time (زمان انتظار)

P₀ P₁ P₂ P₀

20000

20000 - 70

response time (زمان پاسخ)

on submit request (زمان پاسخ) کے بعد (زوج) کے بعد

(output of) (ڈاٹ خروجی) (ڈاٹ خروجی) کے بعد

for time sharing environment

چند (P₀) (P₁) (P₂) (P₃) (P₄) (P₅) (P₆)

5 72

Ques: CPU کی اسکارم کیسے ہے؟ جو کہ اپنی بارہے response time کے لئے مدد کرتا ہے۔

(Q6)

Answer: scheduling (چیزیں جیسے کہ میرے کے لئے

real time

Scheduling Criteria

even CPU i.e. (fairness) \rightarrow (fairness) \rightarrow CPU utilization (Globally Fairness) \rightarrow (latency) overhead

computation resource (جیسا کہ ایسے کام کیلئے کافی ہے)

scheduling

scheduling (جیسا کہ ایسے کام کیلئے کافی ہے اور کام کیلئے کافی ہے)

process

Scheduling Algorithm optimization

criterion

\uparrow CPU utilization = max

\uparrow throughput = max

\downarrow turnaround time = min

\downarrow waiting time = min

\downarrow response time = min

\uparrow fairness = max

\downarrow overhead = min

First come first served | FCFS

scheduling

ex: (process 1, process 2, process 3) \rightarrow (process 1, process 2, process 3)

(process 1, process 2, process 3) \rightarrow (process 1, process 2, process 3)

process 1, process 2, process 3, burst times

process	Burst time
P ₁	24
P ₂	3
P ₃	3

↑
↓
↓

Waiting time = $0 + \Sigma E + \Sigma V = 15$

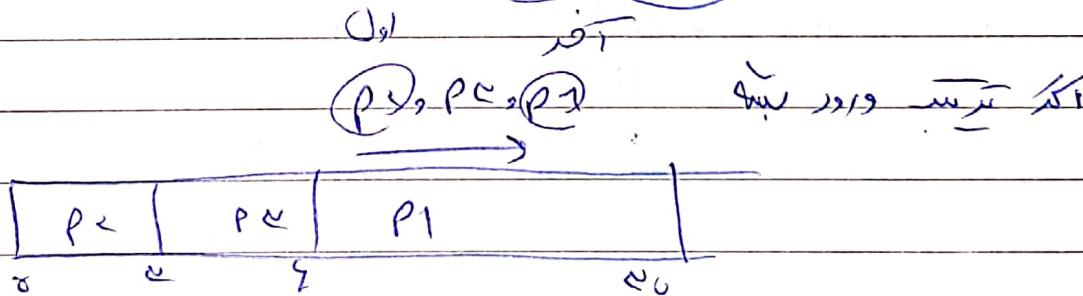
(میانی)

Response time = $0 + \Sigma E + \Sigma V$

برای اولین کاربر (CPU) $\Sigma E + \Sigma V$

$15 =$ ✓

waiting time = response time



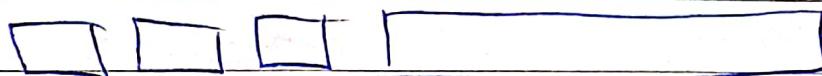
Avg waiting time = $0 + 12 + 4 = 16$

لذیع و میکنند و نیز میگویند این روش FCFS است
اولین کاربر که آغاز کرد اولین کاربر است و این روش
کاربری خوب نیست که اول اولین کاربر سار لر فروخته
که این روش حبیبی نیست که این روش مسأله ای نیست این روش
باید بخوبی راهنمایی شود

فیلم Convoy effect

iii) long burst process \rightarrow convey effect

• IO bound (CPU waiting time) \rightarrow CPU bound (CPU waiting time)



(CPU bound) \rightarrow CPU waiting time \rightarrow IO bound

EX:

	Arrival Time	Completion Time
A	0	3
B	1	3
C	4	3
D	6	2

A	B	C	D	Completion Time
0	3	5	9	11

$$\text{avg waiting time} = \frac{0 + (3-1) + (6-4) + (9-5)}{4}$$

$$= \frac{0 + 2 + 2 + 4}{4} = \frac{8}{4} = 2$$

First CPU works in non preemptive

process \rightarrow No starvation

long burst \rightarrow short burst \rightarrow least overhead

short burst \rightarrow long burst \rightarrow least overhead

FIFO property

اگر تھے ایسی الگوریتم سے پہنچ کر جو بدلی کرنے اسماں کو بروک ہے
صلحتی و نفعی باہم خوب بعلت کرنے چاہیے اگر بروک نہ رکھے اول بدلے
بروک کو چیزیں خوب صدقہ مانند

بڑا لامعا کم باہم \rightarrow burst کہ وہ خوبی

چلتا ایسی الگوریتم را پایا رہ کر.

- a) (1) صلکی مباری کیا جائے کوئی اچھا نہیں
 - b) (2) دعم از اول صرف سہی رائی و ہمیں
- پھر

overhead خوبی

shortest job first (SJF) scheduling

صفہ را فوری بھیج کر جو طبق بروک نہ را اول بدلائیں

burst مبارکہ

avg waiting time

اونچے کر جو اول نہ را اول بدلائیں

SJF

بھی کہنے والے CPN = burst اونچے ایسا

کو ایسی حالت داری.

مین ایجی ویٹنگ تائیم

خوبی بھرنا کوئی رائی

از کجا سائیں کر جو بروک نہ رکھے اول

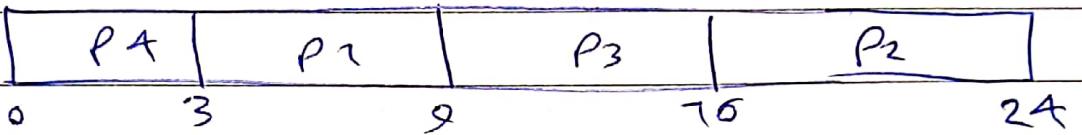
بھی کہنے

ت بھی کہنے کریں اسے

بھی کہنے بروک اسی حالت داری کے بعد burst را دے

process	Burst time	JIF
P1	6	
P2	8	
P3	7	
P4	3	

اگر خرچ کم تر ہے تو کمتر سفر وار (shorter JIF)



$$\text{avg WT} = \frac{0+3+9+16}{4} = \boxed{7}$$

starvation

non preemptive

کوئی کامپیوٹر نے اسی میں کام نہ کر سکتا (non pre-emptive)

waiting time

SJF properties

min avg waiting time

is P3 تاں جس کا بیشتر time ہے اس کا waiting time بھی بڑا ہے

اگر P4 کا burst time 5 ہے تو P3 کا 12 ہے اس کا waiting time 7 ہے

جس کا waiting time 12 ہے اس کا waiting time 12 ہے اس کا waiting time 12 ہے

burst = CPU burst

Exponential averaging (Exponential CPU burst)

① t_n = actual length of nth CPU burst

$$② \tau_{n+1} = \alpha \text{ burst}(S) + (1-\alpha) \tau_n$$

$$③ \alpha, 0 \leq \alpha \leq 1$$

$\oplus \quad \tau_{n+1} = \alpha t_n + (1-\alpha) \tau_n$

$\tau_{n+1} \rightarrow \alpha = \frac{1}{\tau_n}$

(جی burst) $\oplus \text{burst} + (1-\alpha) \tau_n$

جی واقعی burst نیں
جی کو اپنے سامنے بھاگ کر دیں

جی واقعی burst کرنے

جی واقعی burst کو کم کر دیں
shortest remaining time first

SJF (Shortest Job First) Preemptive version

process

arrival time

burst time

p1

0

8 7

p2

7

4

p3

2

$\frac{20}{\Sigma}$

p4

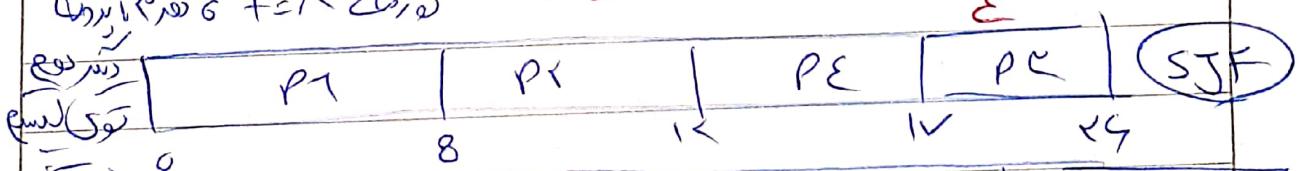
3

5

$\frac{9}{\Sigma}$

avg wts = $\frac{0 + (2-1) + (10-3) + (5-3)}{\Sigma}$

avg wts = $\frac{0 + (2-1) + (10-3) + (5-3)}{\Sigma}$



$$\text{avg wts} = \frac{(10-1) + (7-1) + (17-2) + (5-3)}{\Sigma}$$

(P1) →

$\text{CPU} + 0 \rightarrow \text{Ready} \rightarrow \text{Time}_{\text{units}}$
CPU waiting time

100% \leftarrow 0% waiting time کمتر

(P2) →

$\text{CPU} + 1 \rightarrow \text{Ready} \rightarrow \text{Time}_{\text{units}} + 1$,
CPU waiting time \leftarrow CPU waiting time + 1

(P3) →

$\text{CPU} + 2 \rightarrow \text{Ready} \rightarrow \text{Time}_{\text{units}} + 2$,
 \downarrow

$IV = 100 \rightarrow (100 - 100) / 100 = 0$ \leftarrow 0% CPU time

(P4) →

$\text{CPU} + 5 \rightarrow \text{Ready} \rightarrow \text{Time}_{\text{units}} + 5$,
 $CW = 15$

$$\frac{\text{avg wt}}{\text{wt}} = \frac{9+0+10+5}{5} = \frac{24}{5}$$

Session 17

چند طبقہ کام کی

CPU کا کام کرنے کے لئے CPU کو کوئی از نہیں کروں گے

time quantum

q

Round Robin
(RR)

100, 10, 80

q

ready queue

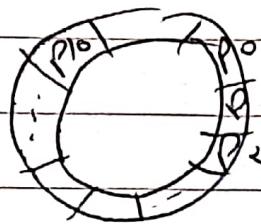
ready queue

time quantum \rightarrow ready queue

q



hirmandpaper
of time units



بعد از P_{10}, P_{10}, P_{10}

اصحای تو و بعد (وارد)

هر کام از بروجها چهار زیر خانه (four) باز round robin

تکی این الگوریتم یعنی (Round Robin) بازی زمانی مساحت را
که کمترین کمینه (time quantum) برای Q_i که در Q_i کمینه

این زمانی کمینه و بعد از این زمانی تا Q_i CPU ایز Q_i ایز

بروچهای (Round Robin) تکی این الگوریتم

FIFO برروجها را تواند صدای ب صورت

باند ایز Q_i کمینه ایز Q_i ایز

حالات بصری کمینه بروجها

CPU ایز Q_i بروجها را ب صورت

هر یک

فعال (Active) base Round robin

متقدی کمینه

برروجها با کوئی زمانی و دلایلی نباشند

$\frac{1}{n} = CPU$ ایز

max response time

کمینه (Minimum) (Maximun) ایز Q_i بروجها

(Minimum) Q_i

کمینه (Minimum) حکم رسانی

بایک

SIVD (SIVD)

الغایه کمینه timer interrupt

hirmandpaper

از Q_i و Q_j کمینه کمینه کمینه

۱

برنورمیون (RR)

با پردازشگر بارگیری قدرت را برای Performance RR

FIFO

بروکس زمانی burst بین اینها نداریم و قبلاً در آنها می‌باشد

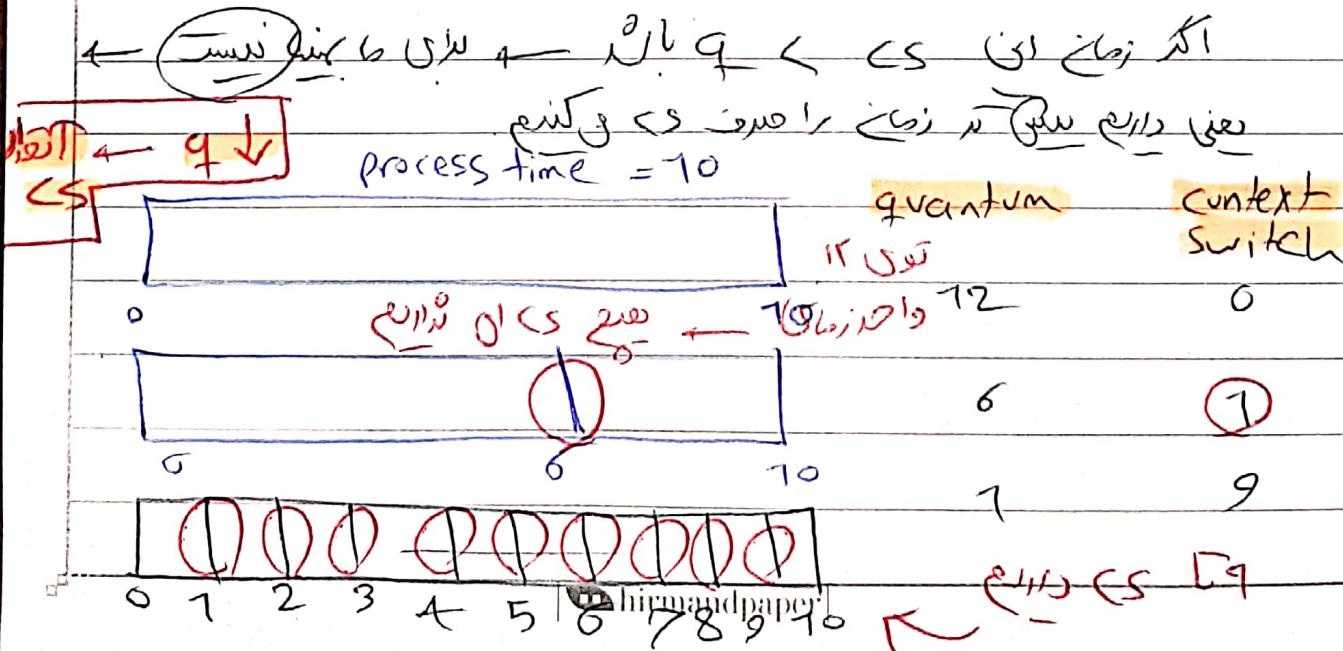
اگر قبلاً در خلی بزرگ بسیم ← بروکس زمانی همان را باز نهاده کنیم و
FIFO را از پس داشته باشیم ← کار رفته را کنم ← می‌توانیم اینجا را باز نهاده کنیم
که متفاوت است و مقداری کمتر از اینجا بروکس زمانی را داشتیم

قبل از آنها burst بروکس زمانی را کوچک بسیم ← اگر قبلاً در خلی بزرگ بسیم ←
CPV را از پس گرفته و → داریم بین بروکس زمانی باز نهاده کنیم
بله ادن بروکس زمانی دوباره لازمه است (صلح) داریم
بروکس زمانی تا جایی که می‌توانیم (context switch CS) → بروکس زمانی را تا جایی که می‌توانیم (context switch CS) →

این همین زمانی می‌گذرد

overhead

CS



نہیں
کوئی 100ms تا 10ms ہے اور $Q = \frac{1}{T}$

context switch < 10 usec

بائی سسٹم نے

ایک بار کر کے سکتا ہے اور جو دوسرے کے لئے مکمل ہے

بائی خلیے پر کے لیے خلیے کو

$$q = \epsilon \quad ! \quad RR \text{ is ex}$$

process burst time

P1

24

P2

3

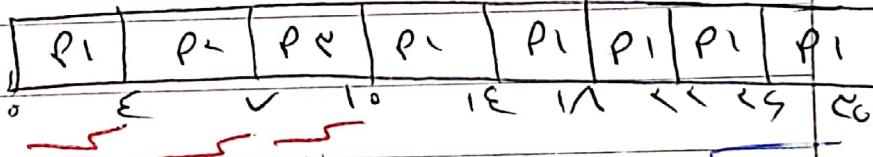
P3

3

ϵ چھٹا

ϵ چھٹا

Gantt chart



(جیسے وادی ϵ کے بعد ϵ کے بعد P_1 کے بعد)

average waiting time

الوقتیں

turnaround time

response time

(Turnaround SJF)

avg waiting time \leftarrow shortest job first

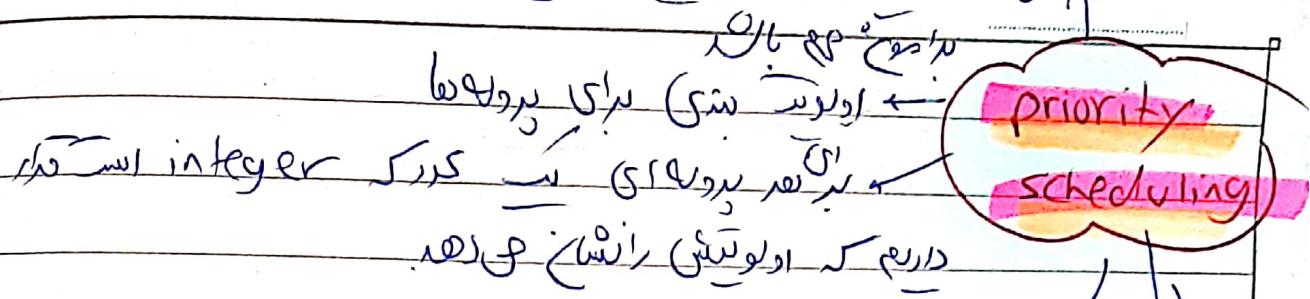
\rightarrow SJF is \leftarrow RR ; wt

RR \rightarrow SJF \leftarrow turnaround

میں کوئی RR گھنیں UNRR ← response time
 (Q RR کوئی CPU میں ورک کرنے والے رکھ دیں) ←
 نوبت فریزہ اور اپنے ساتھ رکھ دیں کاربر
 response time → UNRR ← response time
 time رکھ دیں، response time

SJF (جی ایف) کے ایسے زمانے ہیں
SJF نو تک RR (جی ایف) کو کمینہ نہیں کر سکتے
 قبضہ کرنا
 (non pre-emptive) preemptive ① ← RR (جی ایف)
 starvation ② ← کارل انہیں کسی
 بروڈکاٹ کو پہنچانے کے لئے
 time sharing ③ ← ایسے زمانے ہیں کہ کمینہ جی ایف کا
 راجعہ نہیں ④ ← ایسے زمانے ہیں کہ کمینہ جی ایف کا
 low throughput ← کمینہ جی ایف کا ← ایسے زمانے ہیں کہ کمینہ جی ایف کا
 JLFCS (جی ایف ال ای ایف سی ای ای) ←

بروکسیکی سینچ نامی اولویت باری داشته باشد یا نه



جیف الکاتر این اولویت را برای که CPU نه بروکسیکی که اولویت را برای که CPU

0 انتظار دارد و دور

اوی که اولویت باری دارد

درک همچنان که نه مدفع

CPU رستی باری

① preemptive صورت

② nonpreemptive توانیم بزرگ شروع

اس سینچ priority scheduling SJF
base

اون برداشتی که burst هایی که اولویت دارند زور

بیشتر و اولویت باری دارند

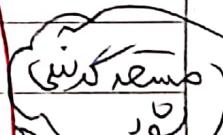
این

starvation

صلل ای التعریف

این برداشتی که اولویت کم دارند همیشه انتظار دارند

نه تنفس



این اتفاقاً به preemptive میگذرد

این اینستور است

P1 صادر و اولویت باری دارد

P2 نیست

برداشتی اولویت باری دارد

اوی

Aging

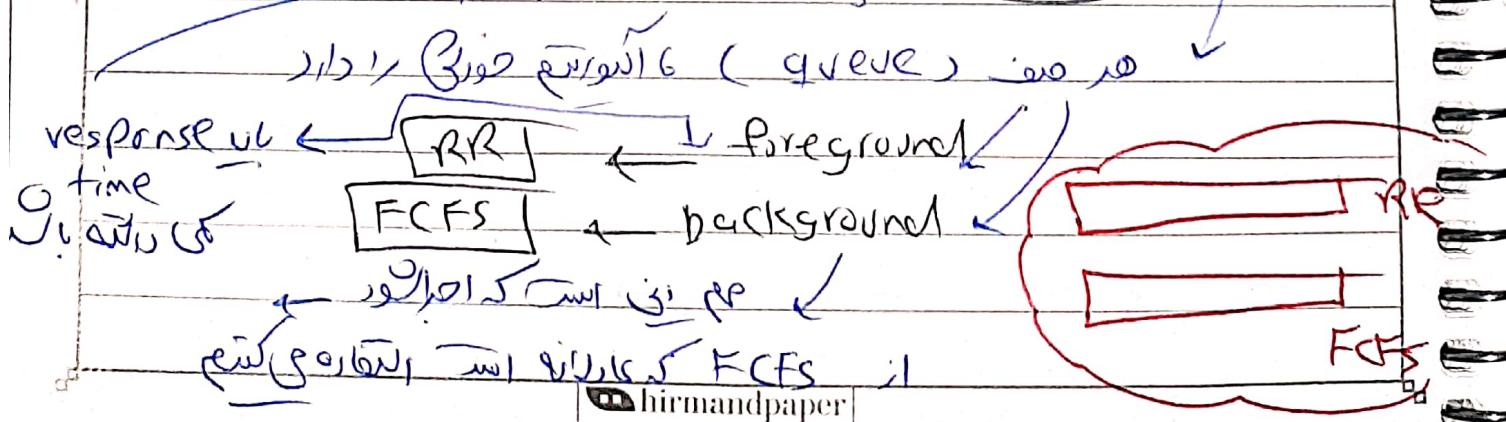
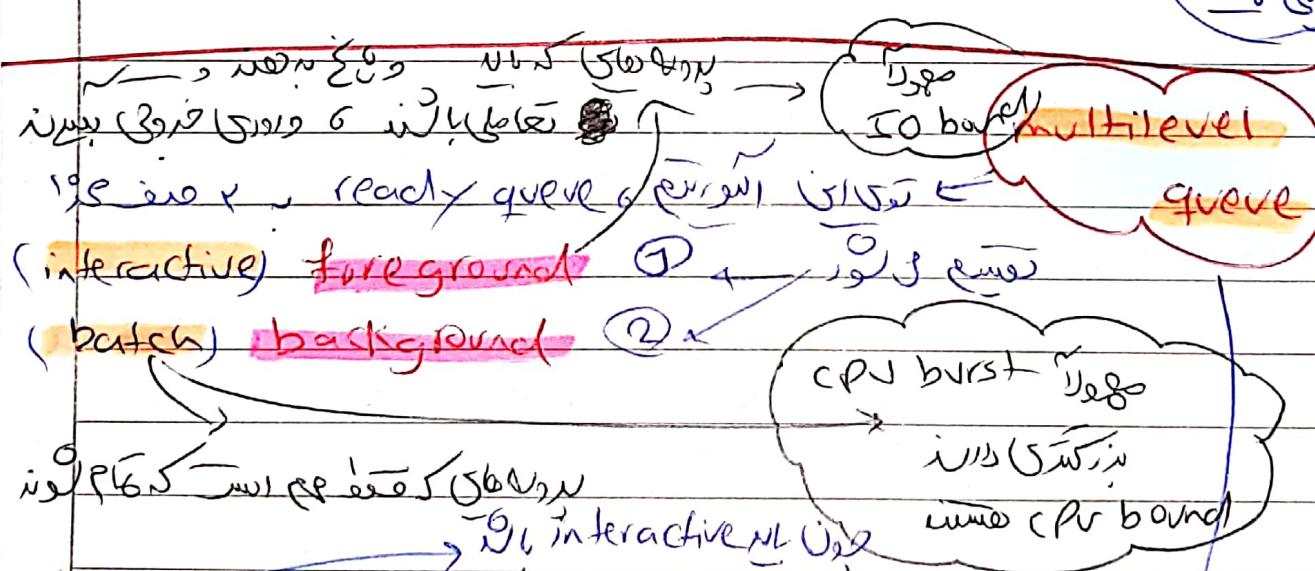
process	Burst Time	priority
P1	10	7(3)
P2	1	1(1)
P3	2	4(4)
P4	1	5(5)
P5	5	2(2)

الخطوة الأولى (1) : ادوات الـ Count Chart

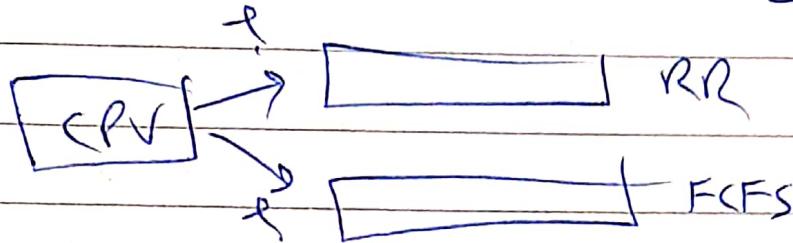
P1	P2	P3	P1	P2	P3
0	1	?	14	15	16

$$\text{avg wt} = \frac{0 + 1 + 4 + 14 + 16}{5} = \frac{\sum_i}{5} \text{ ms}$$

avg wt (avg waiting time)



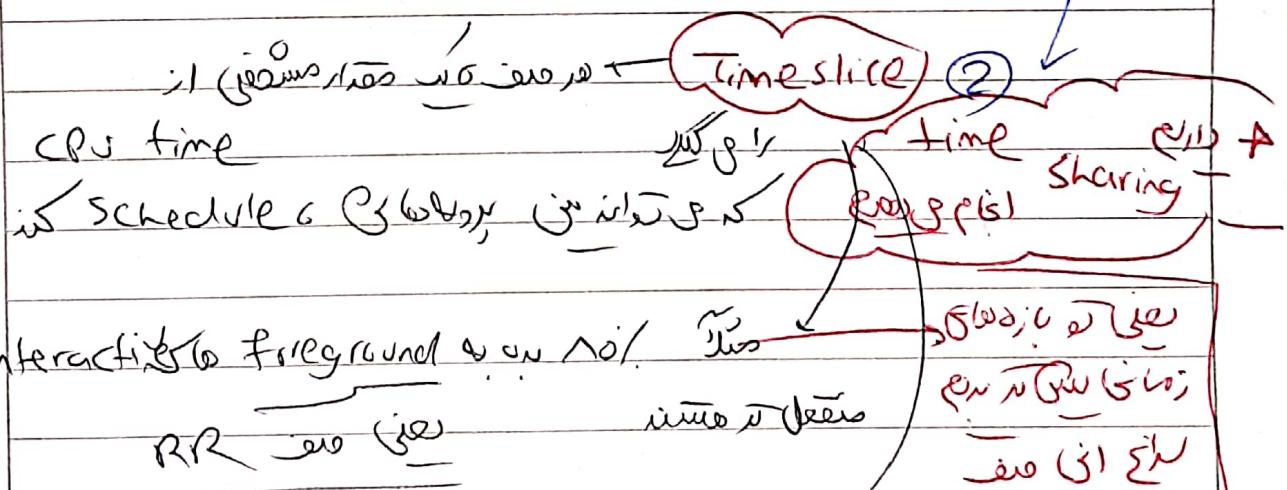
وچ قاعده می باشد که این CPU را در نظر نماییم و از این قاعده برخورداری کنیم



ساده ساختار شرکت خود را در میان این دو داشتیم

۱) scheduling با ثابت priority (Fixed priority) ①
۲) background (پس زمینه) ۳) foreground (پیش زمینه)

۴) FCFS ✓ ۵) starvation (کسری) ✗



FCFS ۱) background ۲) foreground

اولین

real time process

system process

interactive process

batch process

multilevel queue scheduling

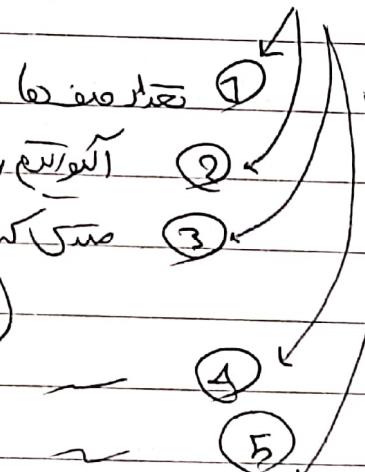
Multilevel queue scheduling \rightarrow آنوریم که برای آنلاین و بروز رسانی است

Multilevel + Feedback Queue

background task \rightarrow فریق که برای گذشتیم (وک) \rightarrow CPU burst
 I/O bound task \rightarrow بعد از این جایی بینهایت (وک) \rightarrow CPU burst
 short task \rightarrow کوتاه دارک ساده بازگردان \rightarrow CPU burst
 long task \rightarrow طولانی \rightarrow CPU burst
 feedback loop \rightarrow از اینجا \rightarrow نیاز به تغیر کردن در قاعده زمانی \rightarrow تغییر کردن در قاعده زمانی

از اینجا برای کسری Feedback loop

برای اینکه بروز رسانی صفت خوبی داشته باشد \rightarrow (برای اینکه همه کارها را در یک زمان محدود انجام دهند)



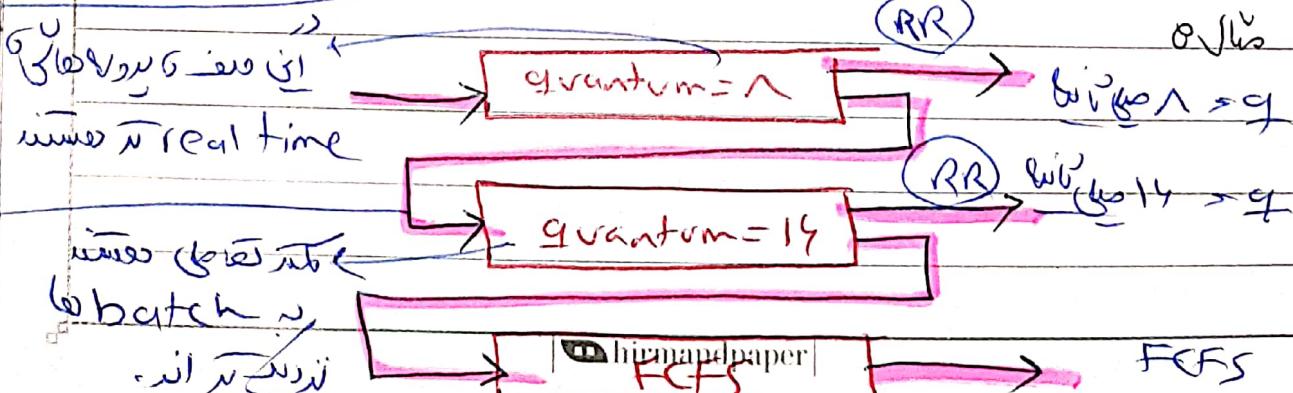
scheduling rule (آنوریم که برای کدام کار را اخراج کنیم)

priority scheduling (برای کارهای کمتر اولویت دار کنیم)

round robin scheduling (برای کارهای هم زمانی که همچنان اولویت دارند)

short job scheduling (برای کارهای کوتاه دار)

long job scheduling (برای کارهای طولانی)



انی جا انی مسلک حل لد (با چه اول کردن یعنی راه ریزی کاری که تا وقیع هدایت می شود)

بره جایی که تعلق دارد (۵)

cheduling (روزی)

گفت طور حسنه واریشنگ می شود و اولین بدهی تویی و بازه زمانی احتمالی فی کنین

صفحه ۲۰

burst بعد از آنها می تانند کام
خواهی لور

صفحه ۲۰ صفحه ۱۹

ب (اندازه) CPU ۹ = ۱۶ ب (اندازه) CPU

مکل اثوارهای می (نی) خواهد ای اسکھن کانسٹر که بروایتی واریلور و از طریق burst ۱

بعد

burst

صفحه ۲۰ صفحه ۱۹

از ۱۶ بورن)

feed back

اوپرس این برد می از

ماشین پاس

احبیل لور بیانی ب می طان

نهیمه معنی اولی

درجه احتمالی

مکل ای راهی

باقراری (۴) می بآفرارک کل CPU را سیر

مکل ای راهی

این بودیلیکی که تویی صفحه اول احتمالی و نیز interactive

bound ۱۰ بورن

burst ۱۰ وی interactive + صفحه ۱۹ اکثر احتمالی لور

بررس

معنی لور بورن > burst

صفحه ۱۹ که غیر راسنگ نیز دفعه کریم نیز نیز

خوبی ممکن است → این به دورتاد خاص و آنچه عرضه شده (۱۵) هست

خوبی داشته باشند

و این RP خوبی که از آن برای خوبی داشته باشند

RP برای response time

realtime

scheduling

soft realtime
systems

(۱۶) نیز

realtime

hard realtime
systems

Scheduling realtime

زمانی از زمان

پیش

مسئل خاصی از تاخیر
سینه کنار

برای انجام کارایی درایی را

از این دلایی گذشت که

(۱۷) نسبت بین خانه های زوار

که دلیل استعفای پروران برای دوستی های نجیب را

توک در اینجا (۱۸) آنچه نیز اینچه خواهد شد

دلایی خوبی
نمود

(P value) Periodic

①

event

(+) processing time

②

انداخت

برای real-time

که

که

(d) deadline

③

برای

برای

که

که

که

آنچه را در دستور سرور انجام داد

آنچه را در دستور سرور انجام داد

که

آنچه را در دستور سرور انجام داد

که

آنچه را در دستور سرور انجام داد

که

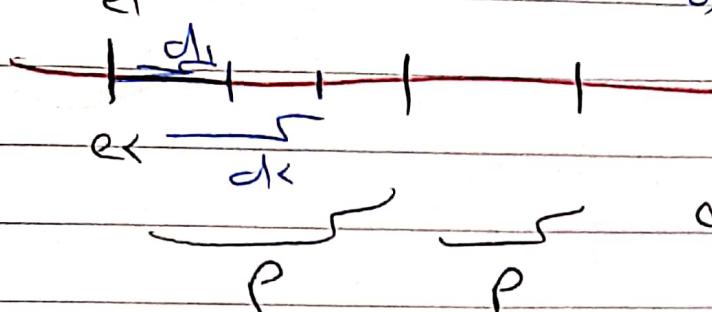
priority based

برای

برای

response time (d) زمانی درین \rightarrow time hard real time

اگر تھی سے کفہ و دکڑوں اور کمپیوٹر پلیسمنی رالائنس (اول اونی) اور
میں کہ درائی مکسیم رارل



بے صدای نہ کرنے

اسے باندھ $(d_1 + t_1 + d_2)$

تھی) اگر بروڈکاول را احمد کسیم، \boxed{P} را بفراہم کی از رائی میں دھرم لے گا

درائی دروں اول $+ t_1 = d_1 + t_1$ میں اسے
زمانی کے قابل کسٹمائزیم فر

Scheduling - جرکھن ایلووئر بروں - Command
↓

chrt(1)

chrt -P 1911V

B (جذب)

برائی دروں سطح ای رکھنے

current scheduling policy

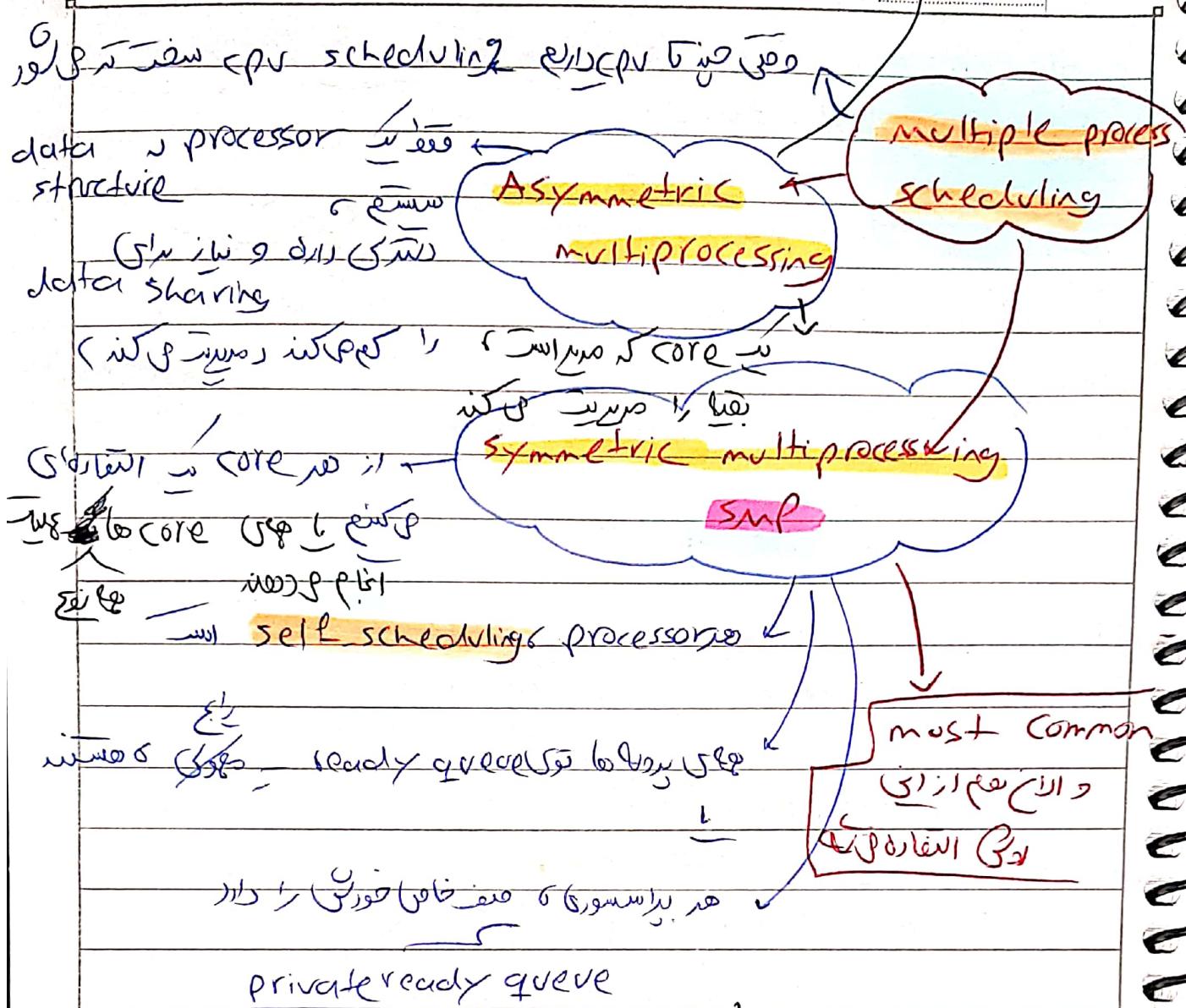
اٹلائی سکی رائے

SCHED_OTHER

current scheduling priority

→ scheduling policy e **sched_FIFO**

ایجاد ایجاد Core یا چیزی که ممکن است task



العنوان \rightarrow المدخلات CPU ; الذاكرة \rightarrow الذاكرة memory stall

one core \rightarrow one thread new row \leftarrow hyperthreading

one chip \rightarrow one thread \leftarrow thread stall

in same physical chip \rightarrow thread stall

chip \rightarrow multiple processor cores \leftarrow multicore processors

in one chip \rightarrow multiple cores

one core \rightarrow multiple threads

multiple threads \rightarrow memory stall

multiple memory retrievers \rightarrow memory stall

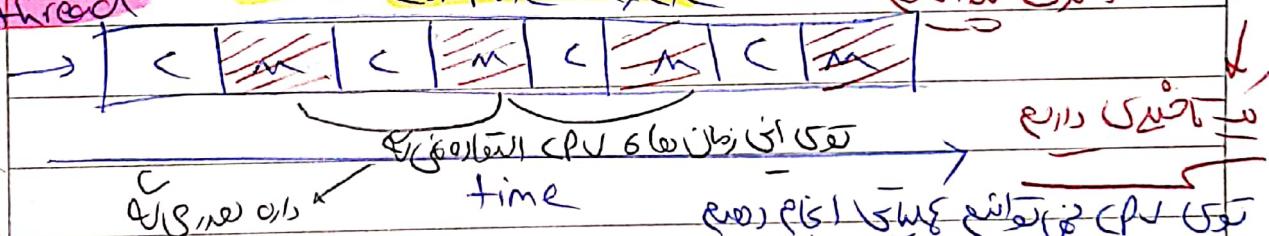
only

memory stall cycle \rightarrow time

compute cycle

M = memory stall cycle

C = compute cycle



one thread \rightarrow one core \rightarrow

one thread

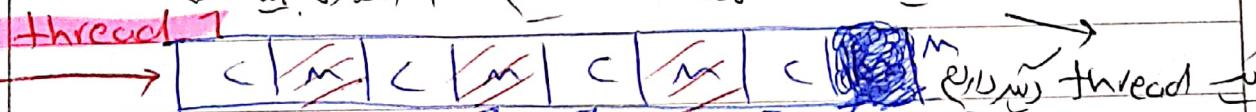
multithreaded

multiple memory stalls \rightarrow multiple threads \rightarrow multithreaded

switching threads

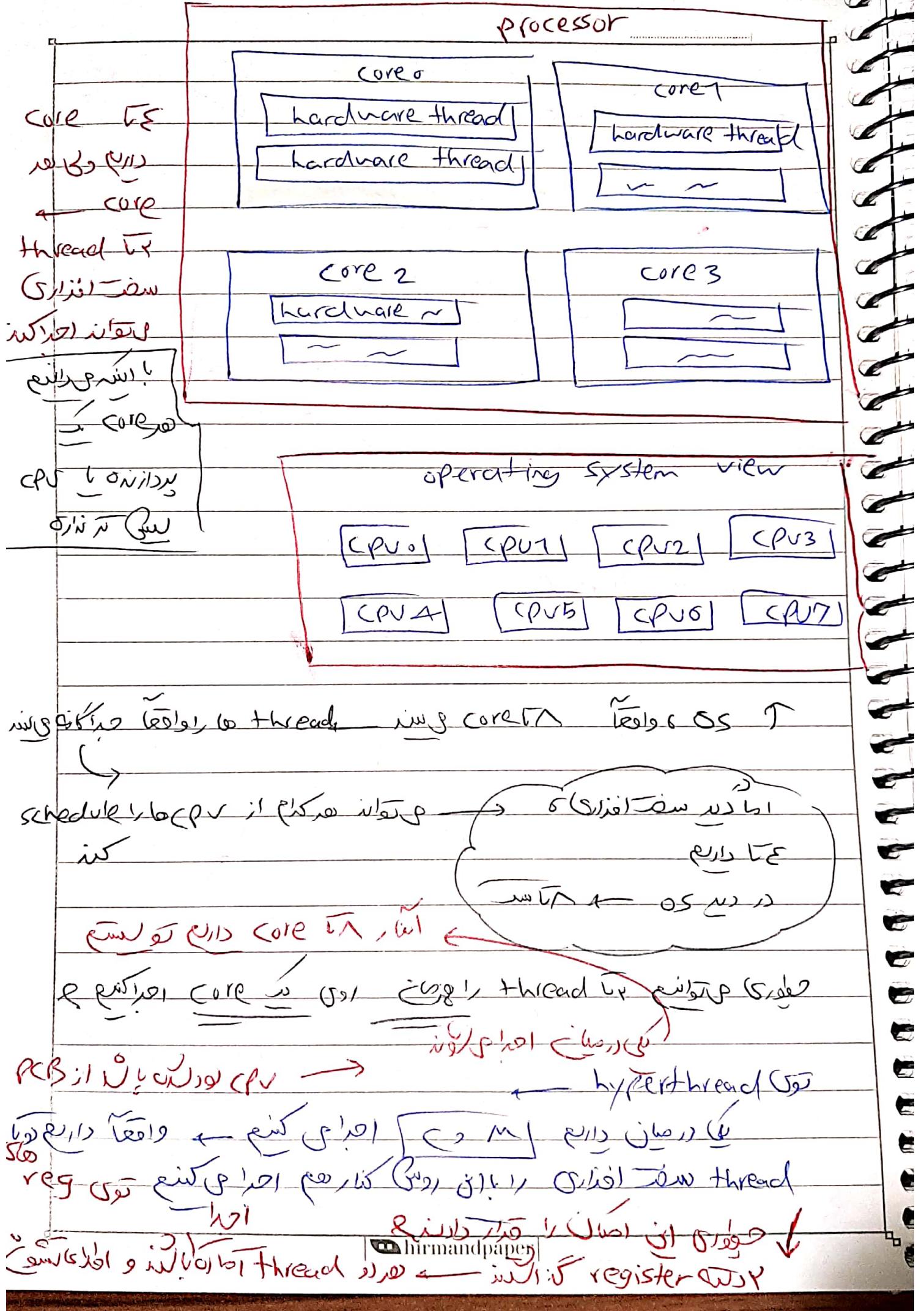
multicore system

one for stall + one stall \rightarrow one thread



time

hirmandpaper



chip multithreading

CMT

assigns software thread to core

is called hyperthreading

by intel

scheduling

logical

CPU has software thread

hardware

maps to core

hardware thread

physical core

software threads

level 1

hardware threads

level 2

processing core

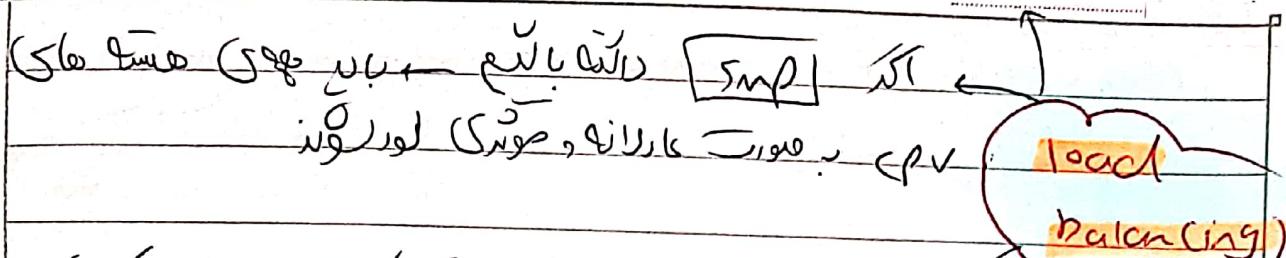
Glob map

Kernel map

Glob map

hardware core

این CPU - core یعنی کدام

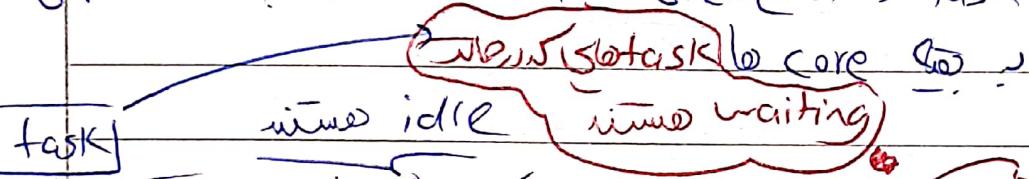


این کاری که local تینه بود local balancing
و core یعنی کدام

برای periodic task push migration

روزانه کار کر که کدام core دارد

که push گردید over load که یعنی core یعنی task



لطفاً این اسکرین شات را اسکن کنید خواسته ام

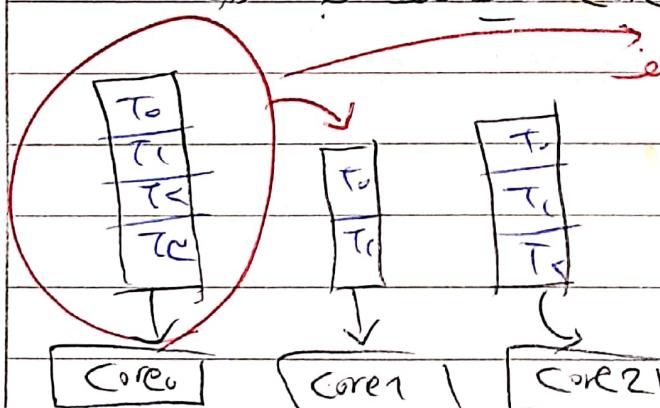
time busy که core یعنی از این خواسته های خوبی خواهد داشت

pull
migration

این کاری که core یعنی که SMP چی

این کاری که task چی

این کاری که core چی



که کدام

که کدام affinity

این کاری

این processor که کدام داشت و میتواند کدام

که کدام داشت و میتواند کدام داشت

که کدام cache که کدام داشت

که کدام cache

taskset - affinity - حکم کردن برای سیفکس هایی - command

برای این سورجها را صدم با افرادی که میگیرند - cache ← affinity
که درینه (وی) ایده ای میگویند پس از آن thread

وابستگی

Cache ویس دلخواه است از هر کدامیک را

آن را بخوبی دستیاب نماییم

Cache ← Core چون تواند از هر کدامیک را بخوبی دستیاب نماییم

این burst از افرادی که با Core

برای هر کدامیک روشی را خالص این کنم دهیار

این cache از افرادی که توی Core

بوده ایشان خالص ندارد و با این جهات از افرادی که از هر کدامیک

Core باشد cache از Core

replace هنوز Core - Cache تواند از هر کدامیک صد هزار دستیاب نماییم

Core هایی که با هر کدامیک را دستیاب نماییم

core & affinity & local balancing

حکم

جیوپر (وی) داریم میتوانیم میتوانیم

ایام میتوانیم و توانیم کنم برویم، اما (ایم) local balancing

برای هر کدامیک از این روشی را خالص خواهد داشت

Core - حکمی که با این وظایق خواهد بود، خالص خواهد بود

و این کنم کنم

affinity

این (کل) برویم (ایم) این کنم

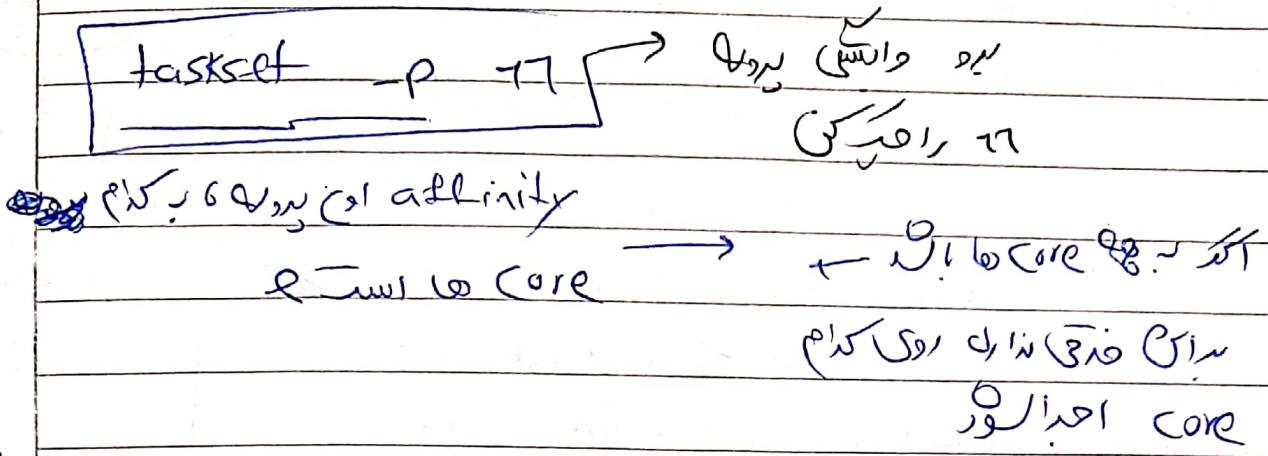
hard affinity

این کنم کنم کنم کنم کنم

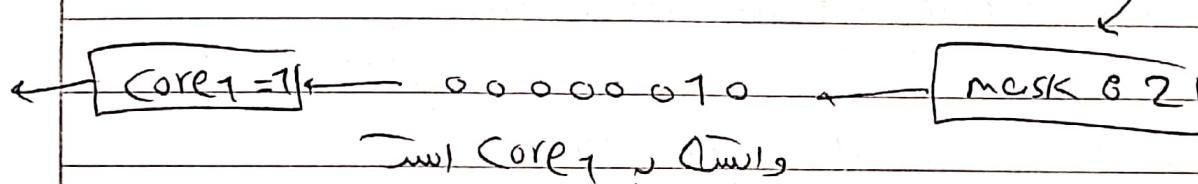
این (کل) برویم (ایم) این کنم کنم کنم کنم

process

اے جو load balancing ہے تو اس کی hard affinity کو اونچی کر سکتے ہیں اسے soft کر سکتے ہیں اسے اونچی کر سکتے ہیں



core 4 پر ہے تو mask کو کام کرنے کے لئے mask کو اونچی کر سکتے ہیں اسے mask کو اونچی کر سکتے ہیں

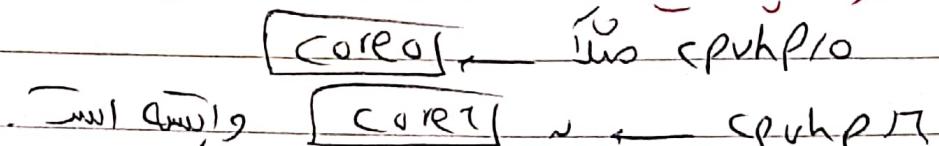


migration ہے جو کوئی core پر ہے migration 1/1 migration 1/0

CPU ہے جو کوئی core پر ہے اسے busy کہا جاتا ہے

balance کو balance کو

کوئی core پر ہے تو کوئی CPU پر ہے تو کوئی CPU پر ہے



(process contention

CPU scope

scope)

(PCS)

SESSION 12

پر سسٹم میں پر سسٹم کا ایک جگہ

in قابل برقرار

kernel

time slices

idle time

idle time

idle time

idle time

idle time

thread scheduling

many-to-many / many-to-one & one thread

پر سسٹم کے بینوں یا PCS scheduler کے طور پر

Thread PCS یا کام کے لئے کام کے لئے thread

kernel کے thread + idle map v. kernel

v. kernel scheduling

virtual CPU

idle core v. kernel scheduling

idle map v. core

idle core scheduling

idle core scheduling

idle core scheduling

idle core

hardware thread process contention

scope

(PCS)

schedule v. core - scope

idle core

System contention scope SCS

این اتفاق را که می خواهیم بخواهیم کنیم را scope نامیدیم
scope که contention scope نامیدیم SCOPE نامیدیم

Scheduler که contention scope دارد SCOPE ①
SCOPE که Scheduler دارد Scheduler ②
cpu که Scheduler دارد Scheduler ③

PCS که scope ای Scheduler و POSIX - API دارد

PCS و SCS هست

POSIX Thread UP PCS و SCS scope دارند

Pthread_scope_process

Scheduler PCS و POSIX دارند

Pthread_scope_system

SCS

processes between

این Pthread_scope_system بین Mac OS و Linux

iii) SCS

OTHER RR و FIFO

other

birmandpaper

کی کی کی scheduling پر ہے جسے scheduling policy

Cpus

chrt | Shell Command

Linux scheduling

pthread_attr_init(&attr)

attr (in's scheduling scope)

in

Config scope

SCS

pthread_attr_getScope(&attr, &scope)

pthread_attr_setScope(&attr, pthread_scope_process)

PCS

CS, PS, User scope

scheduling پر ہے

pthread_attr_getschedpolicy(&attr, &policy)
~ ~ set_sched_policy(&attr, sched_Fifo)

one-to-one ← تعلیمیں

set_scope scheduling پر تعلیمیں

کیون ادھار ملے جائیں (جس کوں رہیں ہیں)

one-to-one میں ملے جائیں (جس کوں رہیں ہیں)

کوں رہیں ہیں (جس کوں رہیں ہیں) کوں رہیں ہیں

one-to-one scope کیوں + کوں رہیں ہیں

کوں رہیں ہیں (جس کوں رہیں ہیں)

one-to-one

one-to-one کیوں کیوں کیوں کیوں کیوں کیوں کیوں کیوں

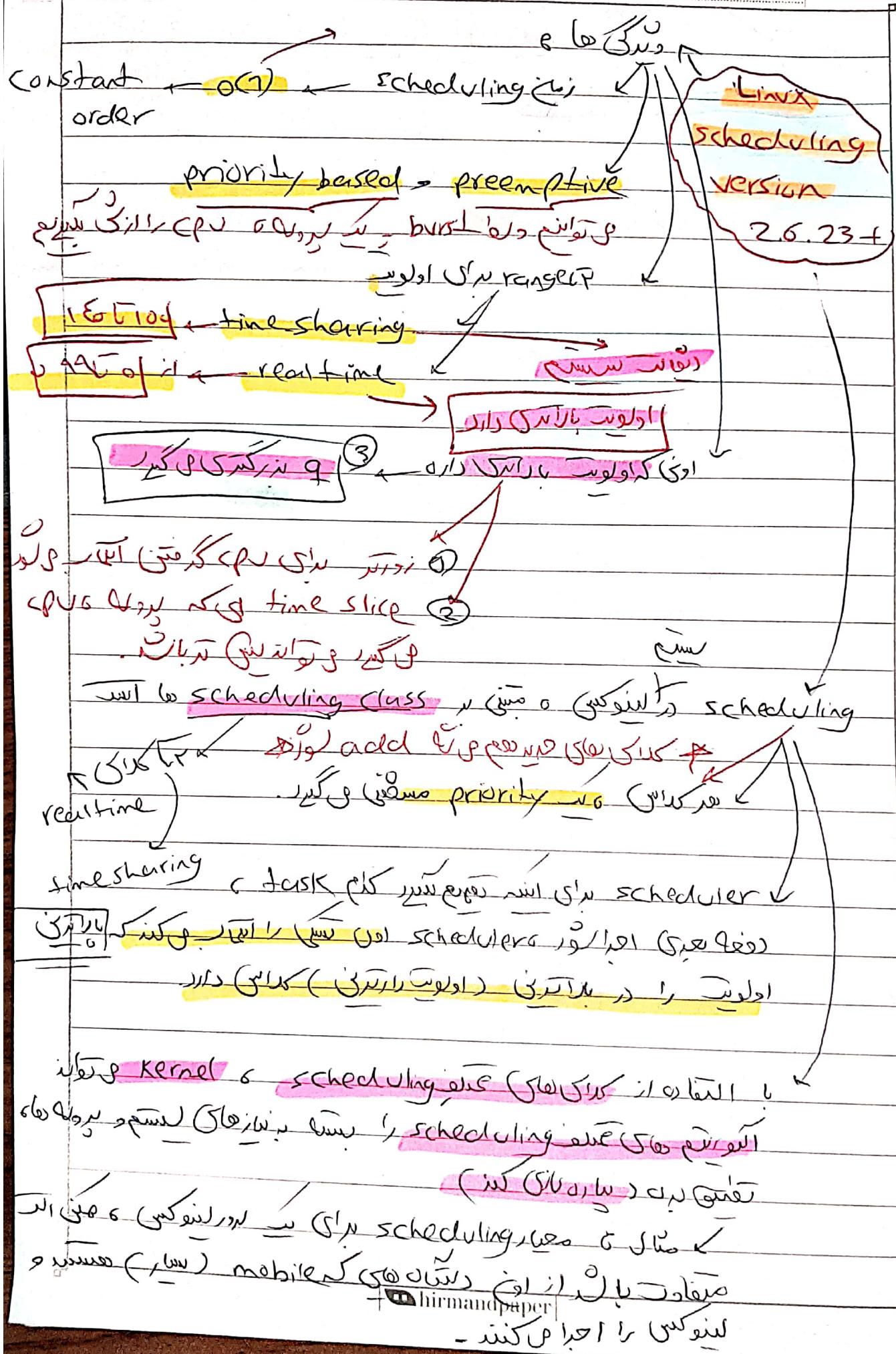
one-to-one

کیوں

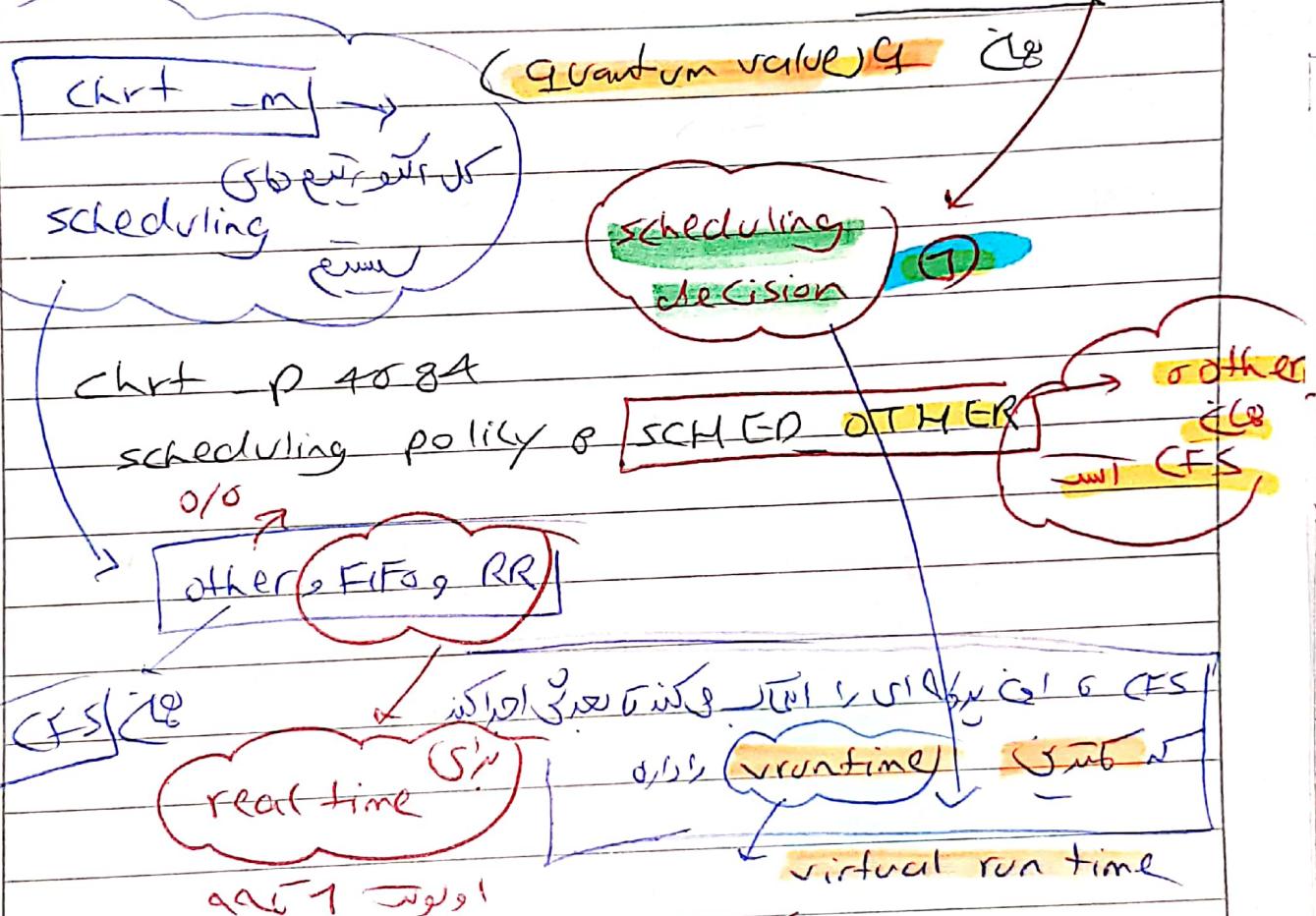
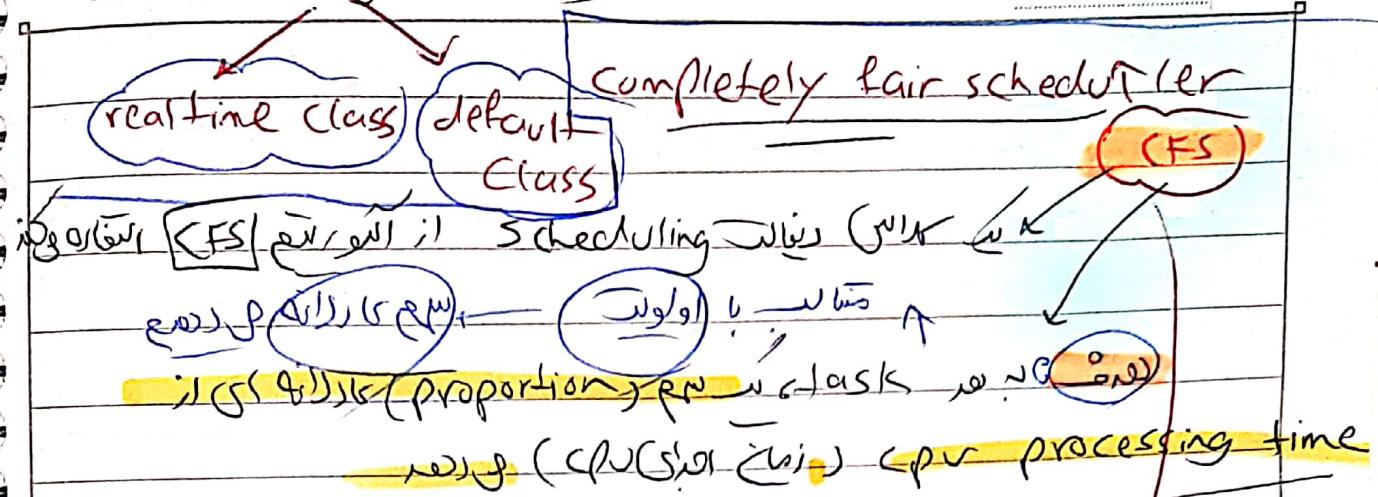
sharmadpaper

Q19 PCS

کامپیوٹر کے کام کے اور بیوپیو کے سامنے کی کامیابی کی تحریر



in P(S) OJW, scheduling (جداول ترتيب) \rightarrow (نحوه) بين kernels



$$P_1 \quad P_2 \\ 20\% \quad 80\%$$

default scheduling \rightarrow time sharing

\rightarrow scheduling other

سازند را نمی تواند بروای که درین مدل
خطی تغیرات و بروای را درین مدل

کاملاً برای این سیستم بازه زمانی را تعیین کنیم

اگر توکن بازه زمانی $T_{context}$ باشد $n = \frac{T}{T_{context}}$
کوچکترین بخش CPU که نیازی نداشته باشد n_{idle}
اجام دفعه n_{idle} باشند

اما این بازه زمانی ۸۰۰ میلی ثانی

هر یک بازه زمانی n بروای را برای P_1 و P_2 می داریم
برای این n مقدار $\frac{P_1}{n} + \frac{P_2}{n}$ بازه زمانی

که بروای را انتخاب کنیم ①
ازین بروای در کنفرانس
البررسی باشند ۲۰٪
تقریباً

هر قدر که بروای کریم ۲۰٪

وقتی که این قدر باشند

schedulers را بروای کنیم

وقتی که بروای کنیم

نحوی سیم باشند (نحوی)

و نحوی و task

و CPU processing time

CPU

2
scheduling
quantum

time slice

انفصال پرینک nice, nice گھرست

پرینک priority

نیک ویلیو nice value

اوی داری با اسکری داری نیک اسکری

Nice value

کا اوی کر نیک ویلیو کار نیک ویلیو

کم کردن سیکل time سیکل time

پرینک CPU

نیک ویلیو سیکل time

کم کردن CPU processing time

کم کردن CPU nice value

نیک ویلیو task اسکری

اوی کر اسکری داری داری task اسکری

نیک ویلیو task را زیارت کن (اوی میرصدرا کا 15)

nice value

دراد نسبت بیغونه

کم کردن اوی میرصدرا کا 15

schedule task کر دوباره task

(بیغونه quantum time

sched latency

an interval of time

مقدار بازه زمانی است که ما می‌خواهیم (وقتی همکاری

targeted latency

برآوردهایی که توانیم آنرا انجام دهیم

latency

برآوردهایی که

آنقدر بود که بازه زمانی را باید نیز بخواهیم

نه خوب نیز
نه فیکی کوچک

اوی میرصدرا کا 15 CPU time است

اوی میرصدرا کا 15 sched latency

اوی میرصدرا کا 15 context switch

نه خوب نیز
نه فیکی کوچک

اوی میرصدرا کا 15

min granularity

کمتر خوارکن

کمتر خوارکن

birmandpaper

اوی میرصدرا کا 15

کارگردانی

min granularity ایسا چیز است که در روزهای زیاد برای time slice است
برای CPU برای min granularity بزرگتر نباشد و بزرگتر باشد.

برای CPU چند جمی $\leftarrow 1\text{ms}$ تا $\leftarrow 1\text{ms}$

virtual run time یا CFS schedule
virtual runtime یا variable time

maintains CPU حالت فعلی
schedule احتمالات اینها را فرموده تردد
خواست داشته باشد که اینها را انتظار کنند
برای اینکه اینها را کنترل کنند

این چنان است که
این را رندرینگ

این خاصیت این است که برای برخی از اینها

task priority

normal default priority

این اولویت دارد که زمانی زیادی داشته باشد

virtual run time = actual run time

nice value اولویت باری داشته باشد
این اولویت دارد که زمانی زیادی داشته باشد

virtual run time < actual run time
virtual > actual

Priority \rightarrow nice value

1. CPU time \rightarrow nice value \rightarrow priority \rightarrow virtual runtime \rightarrow CFS & virtual runtime

time = $T + \sum T_i + \text{idle time}$

$$\text{time-slice}_k = \frac{\text{weight}_k}{\sum_{i=0}^{n-1} \text{weight}_i} \cdot \text{sched latency}$$

run time \rightarrow time interval

$$\text{vruntime}_i = \text{vruntime}_{i-1} + \frac{\text{weights}}{\text{weight}_i} \cdot \text{actual runtime}$$

Overhead \rightarrow virtual runtime \rightarrow scheduler \rightarrow actual runtime

nice value \rightarrow time slice \rightarrow priority \rightarrow run time

Run time \rightarrow nice value \rightarrow weight

$$\text{virtual runtime} \leftarrow \text{vrun time} - \text{weight}_i$$

$$\text{weight}_i = \text{nice value} \cdot \text{CFS}$$

static const int prio_to_weight[40]

Priority weight

Static weight

Priority \rightarrow static weight

Nice \rightarrow nice weight

and nice \rightarrow nice value

Run time \rightarrow weight

(Run time \rightarrow nice value \rightarrow weight) \rightarrow weight

(Run time \rightarrow nice value \rightarrow weight) \rightarrow time interval

Run time \rightarrow vruntime

$n_{vruntime}$ ← n_{Qw_weight} ← n_{nice}

$n_{Qw} = \text{time slice}$

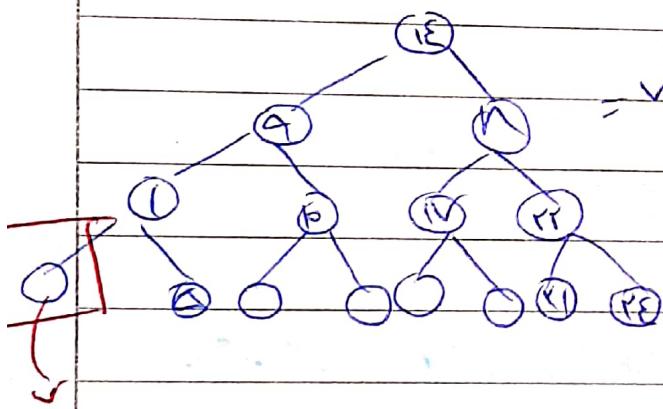
$n_{Qw_vruntime}$ ← n_{weight} ← n_{Qw_nice}

n_{weight} , time slice

الآن $n_{vruntime}$ ينبع من n_{weight} في $n_{Qw_vruntime}$
 $O(1)$

(red black tree)

الآن $n_{vruntime}$ ينبع من $n_{Qw_vruntime}$



$n_{vruntime} = \text{base_unit}$

الآن

$n_{vruntime}$ ينبع من $n_{vruntime}$

الآن $n_{vruntime}$

الآن $n_{vruntime}$ ينبع من $n_{vruntime}$

$O(\log n)$

جذور في الرسم في المنهج

صيغة $n_{vruntime} = \text{base_unit} + \text{الجذور كمئون}$

تم صياغة الجذور بـ $O(1)$

time slice ينبع من $n_{vruntime}$ اي $n_{vruntime}$ هو جذور في المنهج

الآن $n_{vruntime}$ ينبع من $n_{Qw_vruntime}$ ← $O(1)$ $n_{vruntime}$ ينبع من $n_{Qw_vruntime}$

الآن $n_{vruntime}$ ينبع من $n_{Qw_vruntime}$ ← $O(1)$ $n_{vruntime}$ ينبع من $n_{Qw_vruntime}$

اگر کسی بروای بارے time slice قبل از آنکه بروای بارے burst و بروای بارے vruntime باشد

آخر \rightarrow بروای \rightarrow PV \rightarrow burst \rightarrow time slice

vruntime باشد تا این دفعه توان خروجی خالی باشد routine

و هر دفعه بروای انجام می‌دهد که وقتی انجام می‌دهد بروای دفعه کسر است

ای برای بروای در حق در نظر بینهایت vruntime داشته باشد

برای کسی که سعید است \rightarrow IO وقتی بروای داشته باشد

برای routine بینهایت بروای داشته باشد وقتی بروای داشته باشد

صلح PV \rightarrow waiting \rightarrow زیرا توی صدر

باخت کهست بروای داشت \rightarrow راه حل

ای بروای که بینهایت \rightarrow IO

min - vruntime \rightarrow min = vruntime

ای بروای \rightarrow min vruntime $\times \times$

اگر توی لست سالیانه \rightarrow IO

دلتا \rightarrow 6 با این آنوریم که این \rightarrow اولین باره

XX

SW

R min

IO bound \rightarrow PV \rightarrow burst

برای \rightarrow time slice

قبل از آنکه بروای انجام دهد

برای PV \rightarrow وقتی که سعید است

NS \rightarrow interactive

PV \rightarrow بروای \rightarrow (PV) \rightarrow QN

او بروای \rightarrow او بروای \rightarrow بروای \rightarrow IO bounds \rightarrow XX

burst (رسپل) کی جو کم تر CPU bound رہے اور time slice (اندر میں)، time slice (تکمیل) کی طرف رہے

ویسے کام کرنے والے کاموں کو real time کہا جاتا ہے

real time tasks

run CPU بسٹ

FIFO ①
(Round Robin)

RR ②

sched_FIFO, sched_RR

نئی تری

لئے جائے

Linux scheduling
real time

nonreal time normal tasks کی جو کام real time policy کے لئے کام کرنے والے کاموں کی سیکھیں

global priority map
100

- nice value



RR

RRL_FIFO

CFS



higher

priority

lower

$$PR = \text{RT} + NICE$$

top

$$PR = -1 - \text{real_time}$$

priority

real time

CPS-Scheduling | جدول زمانی CPS

لینکیج کیس + Algorithm evaluation

Deterministic modeling

(جواب ممکن است)

طابعی و بدلی و

متقارن

Queueing models

الوقتی و اینکه کجا کجا می شود

جیز نہیں ممکن

متقارن

Simulations

Scheduling Implementation

kernel

بررسی انتظار،

پریوریتی،

و waiting time

بررسی کیا کجا ممکن

کیا کجا ممکن

میتوان