

程序代写代做 CS 编程辅导

COMP2300/6300

Computer Organisation and Programming



Operating Systems

Dr Charles Martin

Semester 1, 2022



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Admin Time

程序代写代做 CS编程辅导

Little wires and alternatives—see [here](#)



[here](#)

Quiz 2 open!

WeChat: cstutorcs

Midsem feedback: marks out on streams, [stats here](#), detailed feedback tonight.

Assignment Project Exam Help

Assignment 2! Go work on it enough to [ask a question](#)

Email: tutorcs@163.com

(This week I'm working through some old issues on Piazza – thanks for your patience!) QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



WeChat: cstutorcs

Week 10: Operating Systems

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Outline

- what is an OS?
- privilege levels
- processes & scheduling

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

What is an OS?



程序代写代做 CS 编程辅导



WeChat: cstutorcs
OS/2

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



Be
OS



talk

程序代写代做 CS编程辅导

what is an operating system (OS)



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

...it's a virtual machine

offering a more familiar, comfort-

- memory management
- hardware abstraction
- process management
- inter-process communication (IPC)

程序代写代做 CS编程辅导



safer environment for your programs to run in

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

...it's a resource manager 程序代写代做 CS编程辅导

co-ordinating access to hardware resources

- processors
- memory
- mass storage
- communication channels
- devices (timers, GPUs, DSPs, other peripherals.)



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

multiple tasks/processes/programs
QQ: 749589476 for access to these resources!

<https://tutorcs.com>

程序代写代做 CS编程辅导



WeChat: cstutorcs

A brief history of operating systems. Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

1950s-60s: system monitors

程序代写代做 CS编程辅导

IBM 704 mainframe at NACA in 1958

NASA / Public Domain



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

1960s - multi-programming system

程序代写代做 CS 编程辅导

University of Manchester Atlas,



63

Iain MacCallum / CC BY (<https://creativecommons.org/licenses/by/3.0>)

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

1970s - multi-tasking systems

程序代写代做 CS 编程辅导

DEC PDP-11



Stefan_Kögl / CC BY-SA (<http://creativecommons.org/licenses/by-sa/3.0/>)

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

1970s - early workstations...

程序代写代做 CS 编程辅导

Xerox Alto

Joho345 / Public domain

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



1970s-80s: Consumer OSs

程序代写代做 CS编程辅导

Springsgrace / CC BY-SA (<https://creativecommons.org/licenses/by-sa/4.0/>)



creativecommons.org/licenses/by-sa/4.0/

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

1990s+

NeXTStation

程序代写代做 CS 编程辅导



Blake Patterson / CC BY (<https://creativecommons.org/licenses/by/2.0>)

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

A brief history of operating systems(1) 程序代写代做 CS编程辅导

in the beginning: single user, single program, task, serial processing—no OS



- 50s: system monitors/batch processing:
 - the monitor ordered the sequence of jobs and triggered their sequential execution
- 50s-60s: advanced system monitors/batch processing:
 - the monitor handles interrupts and timers
 - first support for memory protection
 - first implementations of privileged instructions (accessible by the monitor only)
- early 60s: multi-programming systems:
 - use the long device I/O delays for switches to other runnable programs
- early 60s: multi-programming, time-sharing systems:
 - assign time-slices to each program and switch regularly

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

A brief history of operating systems(2) 程序代写代做 CS编程辅导

- early 70s: multi-tasking systems – multi user environments resulting in UNIX (and others)
- early 80s: single user, single tasking systems (emphasis on user interface or APIs. MS-DOS, CP/M, MacOS and others first employed ‘small scale’ personal computers).
- mid-80s: Distributed/multiprocessor operating systems - modern UNIX systems (SYSV, BSD)
- late 70s: Workstations starting by porting UNIX or VMS to ‘smaller’ computers.
- 80s: PCs starting with almost none of the classical OS-features and services, but with an user-interface (MacOS) and simple device drivers (MS-DOS)



WeChat: cstutorcs
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

A brief history of operating systems(3) 程序代写代做 CS编程辅导

- last 20 years: evolving and expanding general purpose OSes, like for instance:
 - Solaris (based on SVR4, BSD, and SunOS; pretty much dead now)
 - Linux (open source UNIX re-implementation for x86 processors and others)
 - current Windows (used to be partly based on Windows NT, which is ‘related’ to VMS)
 - MacOS (Mach kernel with BSD Unix and a proprietary user-interface)
- multi-processing is supported by all these OSes to some extent
 - but not (really) suitable for embedded, or real-time systems



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Standard features?

程序代写代做 CS编程辅导

is there a standard set of features for operating systems?
no.



the term *operating system* covers everything from 4 kB microkernels, to > 1 GB installations
of desktop general purpose operating systems

WeChat: cstutorcs
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Minimal set of features?

程序代写代做 CS编程辅导

is there a *minimal* set of features



almost: memory management, process management and inter-process communication/synchronisation would be considered essential in most systems

is there always an explicit operating system?

Assignment Project Exam Help

no: some languages and development systems operate with standalone runtime environments

Email: tutorcs@163.com
QQ: 749389476

<https://tutorcs.com>

Process management

程序代写代做 CS编程辅导

(we'll talk more about this **in a moment**)

basically, this is the task of keeping all those things going all at once...



...while tricking them all into thinking they're the main game

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

talk

程序代写代做 CS编程辅导

what's a task?



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Memory management

程序代写代做 CS编程辅导

remember memory? the OS is responsible for sharing it around

- allocation / deallocation
- virtual memory: logical vs. physical addresses, segments, paging, swapping, etc.
- memory protection (privilege levels, separate virtual memory segments, ...)
- shared memory (for performance, communication, ...)



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Synchronisation/inter-process communication

remember all the **asynchronism** OS is responsible for managing that as well

semaphores, mutexes, condition channels, mailboxes, MPI, etc.



this is tightly coupled to scheduling / task switching!

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Hardware abstraction

程序代写代做 CS编程辅导

remember all the specific load-trick addresses in the labs? no?



good news everyone! the OS does all that for you. you don't have to

- device drivers
- protocols, file systems, networking, everything else...

all through a consistent API

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Kernel: definition

程序代写代做 CS编程辅导

the **kernel** is the program (functions, data structures in memory, etc.) which performs the **core** role(s) of the OS



access to the CPU, memory, peripherals all happens *through* the kernel through a **system call**

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



程序代写代做 CS 编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

if you want to look at some real APIs

on Linux,

- **syscalls.h header file**
- **how to add a new system call**

on Windows,

- **Windows API Index**



APIs

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



writing an OS seems complicated

WeChat: cstutorcs Assignment Project Exam Help

how is it done in practice?

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Monolithic OS

(or ‘the big mess...’)

- non-portable/hard to maintain
- lacks reliability
- all services are in the kernel (on the same privilege level)
- but: may reach high efficiency

e.g.: most early UNIX systems, MS-DOS (80s), Windows (all non-NT based versions) MacOS (until version 9), etc...

程序代写代做 CS编程辅导



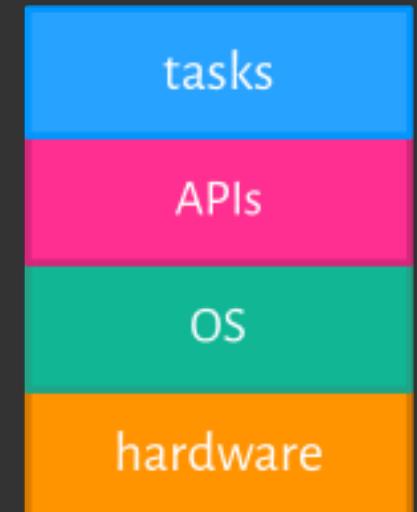
WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



monolithicOS

Monolithic & Modular OS

程序代写代做 CS编程辅导

- modules can be platform independent
- easier to maintain and to develop
- reliability is increased
- all services are still in the kernel (on the same privilege level)
- may reach high efficiency

e.g., current Linux versions



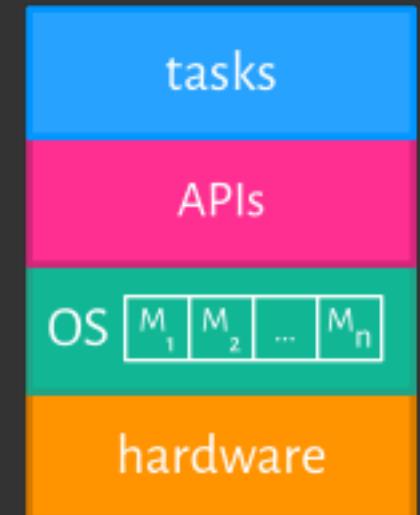
WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



monolithic &
modular OS

μ Kernels & client-server models

程序代写代做 CS 编程辅导

- μ kernel implements essential processes and message handling
- all ‘higher’ services are user level servers
- kernel ensures reliable message passing between clients and servers
- highly modular, flexible & maintainable
- servers can be redundant and easily replaced
- (possibly) reduced efficiency through increased communications

WeChat: cstutorcs

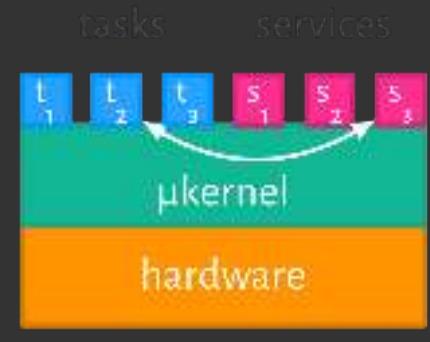
Assignment Project Exam Help

e.g., current research projects, μ L4, Minix 3, etc.

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



THIS IS A UNIX SYSTEM

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

I KNOW THIS

Example: UNIX

程序代写代做 CS编程辅导

- hierarchical file-system (mainly based on `mount` and `umount`)
- universal file-interface applied to files, devices (I/O), as well as IPC
- dynamic process creation via duplication
- choice of shells
- internal structure as well as all APIs are based on C
- relatively high degree of portability



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

many versions/flavours: UNICS, UNIX, BSD, XENIX, System V, QNX, IRIX, SunOS, Ultrix, Sinix, Mach, Plan 9, NeXTSTEP, AIX, HP-UX, Solaris, NetBSD, FreeBSD, Linux, OPENSTEP, OpenBSD, Darwin, QNX/Neutrino, OS X, QNX ROTS, <https://tutorcs.com>

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Privilege levels

talk

程序代写代做 CS编程辅导

what do you think privilege mean?



how does it affect your code running on microbit?

WeChat: cstutorcs

Assignment Project Exam Help

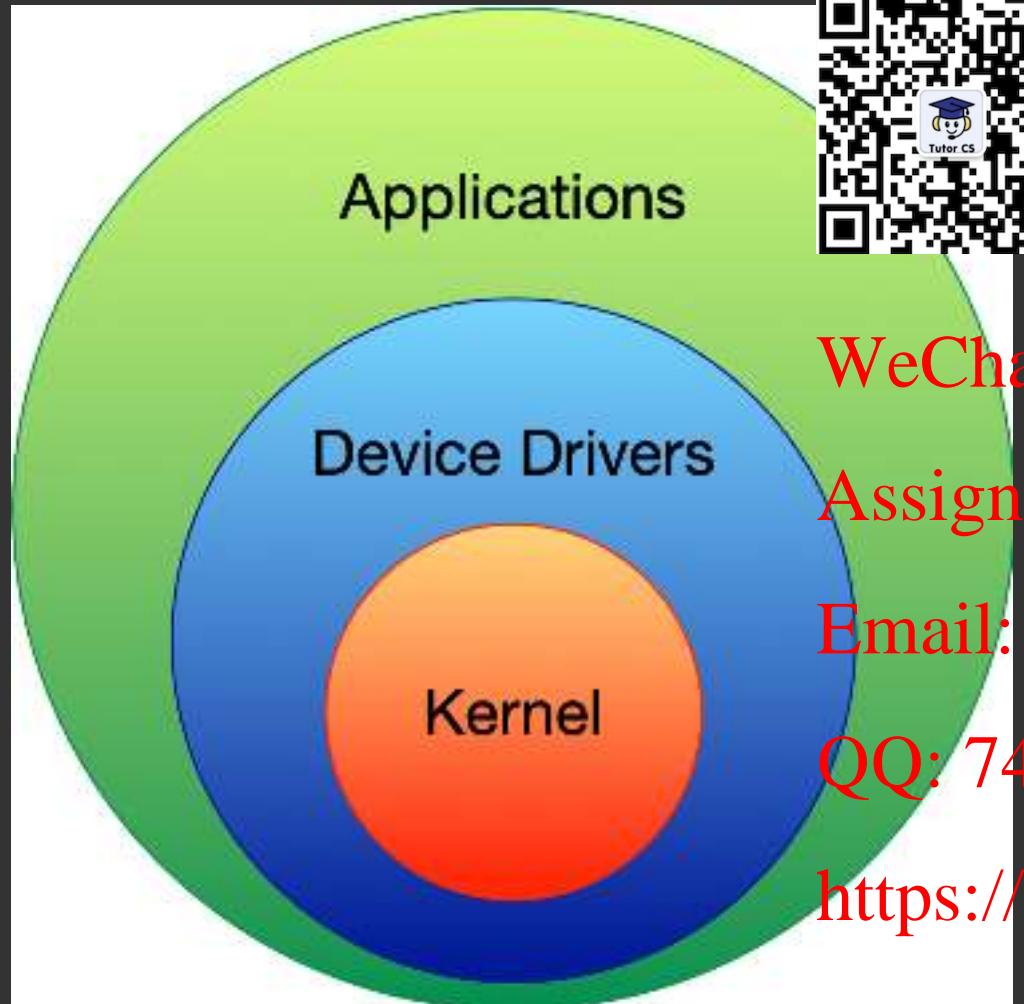
Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Privilege

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Privilege levels

程序代写代做 CS编程辅导

certain instructions can only be executed in a computer's “privileged” mode—this is enforced in **hardware**.
different architectures enforce the privilege separation in different ways



check the manual (e.g. *Section A2.3.4* on p32 or *Table B1-1 Mode* on p568 of the **ARMv7-M reference manual**)

WeChat: cstutorcs

Fun video for the nostalgic: What is DCS protected mode?

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

ARMv7-M execution levels 程序代写代做 CS编程辅导

privileged

thread mode

unprivileged

regular code

regular code



Handler mode

Exceptions (including interrupts)

privileges may control:

- code execution
- memory read/write access
- register access (e.g., for peripherals)

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

“Supervisor call” instruction 程序代写代做 CS编程辅导

have you noticed these entries in the interrupt table in labs?



```
    .word    SVC_Handler  
@ ...  
    .word    PendSV_Handler
```

WeChat: cstutorcs

Assignment Project Exam Help

the `svc` instruction (A7.7.175 in the reference manual) runs the `SVC_Handler` immediately

Email: tutorcs@163.com
QQ: 749389476

Encoding T1
`SVC<c> #<imm8>`

All versions of the Thumb instruction set.
<https://tutorcs.com>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	1	1	1	1								imm8

Deferred supervisor call (PendsV) 程序代写代做 CS编程辅导

there's a **PENDSVSET** bit (bit 28)



in the Interrupt Control and State Register (**ICSR**)

if set, the **PendSV_Handler** will be called according to the usual interrupt/exception priority rules
WeChat: cstutorcs

both **SVC_Handler** and **PendSV_Handler** run in privileged mode, like *all* interrupts

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

System calls with SVC

How might we implement a sys

程序代写代做 CS编程辅导



a microbit?

How can we get at the argument? (Hint: see Figure B1-3 from the ARM-v7 M architecture reference)

WeChat: cstutorcs

How do system calls work in linux?

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



WeChat: cstutorcs

Types of Operating Systems

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Distributed operating systems



- all CPUs carry a small kernel or a system for communication services
- all other OS services are distributed among available CPUs
- services may migrate
- services can be multiplied in order to guarantee availability (hot stand-by), or to increase throughput (heavy duty servers)

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Real-time operating systems



- fast context switches?

should be fast anyway

- small size?

should be small anyway

- quick response to external interrupts?

not *quick*, but predictable

WeChat: cstutorcs

- multitasking?

often, not always

- 'low level' programming interfaces?

Assignment Project Exam Help

needed in many operating systems

- interprocess communication tools?

Email: tutorcs@163.com

needed in almost all operating systems

- high processor utilisation?

QQ: 749389476

fault tolerance builds on redundancy

<https://tutorcs.com>

Real-time OS for music?

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



Real-time OS for robots?

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Real-time OS for cars?

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Real-time operating systems need to provide... 程序代写代做CS编程辅导

the **logical correctness** of the results as well as the correctness of the **time**: *what* and *when* are both important



all results are to be delivered **just-in-time**—not too early, not too late.

WeChat: cstutorcs

timing constraints are specified in many different ways... often as a response to **external events** (reactive systems)

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



WeChat: cstutorcs

predictability, not performance!

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Embedded Operating Systems

程序代写代做 CS 编程辅导

usually real-time systems, often real-time systems



very small footprint (often a few kBytes)

WeChat: cstutorcs

none or limited user-interaction

Assignment Project Exam Help

most processors in the world are in embedded systems

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



WeChat: cstutorcs

Processes & scheduling
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

processes: 522 total, 2 running, 2 stuck, 518 sleeping, 2247 threads
 ad Avg: 23.59, 21.19, 11.70 CPU usage: 5.37% user, 22.46% sys, 72.16% idle SharedLibs: 406M resident, 81M data, 118M linkedit.
 mRegions: 125955 total, 4430M resident, 173M private, 3980M shared. PhysMem: 16G used (2673M wired), 173M unused.
 : 6967G vsize, 1371M Framework vsize, 173655(8) swapins, 212436(8) swapouts. Networks: packets: 4673930/4821M in, 2391521/404M out. Disks: 5991776/63G read, 5768484/51G written.

Trust the Process

程序代写代做 CS 编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

ID	COMMAND	%CPU	TIME	#TH	#WQ	#PORT	MEM	PURG	CMPRS	PGRP	PPID	STATE	BOOSTS	%CPU_ME	%CPU_OTHRS	UID	FAULTS	CDW	MSGSENT	MSGRECV	SYSBSD
835	bztransmit	0.0	00:00:03	3	2	23	2608K	0B	0B	56	1	sleeping	*0[1]	0.00000	0.00000	0	3721	155	352	155	763
828	screenancur	0.2	00:00:29	5	3	198	13M	112K	0B	47821	1	sleeping	*0[1]	0.00000	0.00000	501	8783	350	2637+	795+	3697+
827	screenancur	0.5	00:00:29	5	3	1232K	620K	0B	528	528	1	sleeping	*0[1]	0.00000	0.00000	501	16957+	192	3395+	1150+	2282+
823	mdworker	0.0	00:00:07	7	1	132K	0B	0B	47823	1	sleeping	*0[1]	0.00000	0.00000	501	5307	187	642	270	791	
822	mdworker_sha	0.0	00:00:09	3	1	59	4M	0B	0B	47822	1	sleeping	*0[1]	0.00000	0.00000	501	7944	186	644	271	793
821	mdworker_sha	0.0	00:00:08	3	1	59	14M	0B	0B	47823	1	sleeping	*0[1]	0.00000	0.00000	501	7928	187	640	269	791
820	mdworker_sha	0.0	00:00:08	3	1	59	14M	0B	0B	47822	1	sleeping	*0[1]	0.00000	0.00000	501	7907	187	639	269	783
819	top	4.5	00:01:56	1/1	0	32	5952K+	0B	0B	47821	1	sleeping	*0[1]	0.00000	0.00000	0	9201+	91	476842+	238401+	29825+
797	ManagedClient	0.0	00:00:03	2	1	48	1372K	0B	0B	47821	1	sleeping	*0[1]	0.00000	0.00000	501	3538	152	172	58	595
794	corecaptured	0.0	00:00:02	9	1	72	17M	0B	0B	47821	1	sleeping	*0[0]	0.00000	0.00000	0	1767	128	334	166	621
798	mdworker_sha	0.0	00:00:07	3	1	56	3404K	0B	0B	47821	1	sleeping	*0[1]	0.00000	0.00000	501	3764	191	625	270	1343
789	backupd	0.0	00:00:06	2	1	44	1900K	0B	0B	47821	1	sleeping	*0[1]	0.00000	0.00000	0	8155	199	246	89	857
785	aciseposture	0.0	00:00:50	2	1	45	5050K	0B	0B	47821	1	sleeping	*0[1]	0.00000	0.00000	501	3343	394	207	98	9726
782	kcm	0.0	00:00:04	3	3	21	2284K	0B	0B	47778	1	sleeping	*0[1]	0.00000	0.00000	0	2807	137	494	137	1780
778	SCHelper	0.0	00:00:02	2	1	27	832K	0B	0B	47778	1	sleeping	*0[1]	0.00000	0.00000	0	1638	125	281	35	543
776	syncdefaults	0.0	00:00:43	3	1	114	16M	0B	0B	47776	1	sleeping	*0[18]	0.00000	0.00000	501	10436	261	888	491	8551
773	AirPort Base	0.0	00:00:02	3	1	45	1184K	0B	0B	47772	1	sleeping	*0[2]	0.00000	0.00000	501	1831	126	153	62	769
770	ManagedClient	0.0	00:00:43	2	1	68	12M	0B	0B	47769	1	sleeping	*0[1]	0.00000	0.00000	0	7252	218	3879	493	17583
769	mdmclient	0.0	00:00:19	4	2	83	18M	0B	0B	47769	1	sleeping	*0[59]	0.00000	0.00000	501	8748	252	822	745	4508
766	AddressBookS	0.3	00:01:00	8	6	157	18M-	1340K	0B	47766	1	sleeping	*0[58]	0.00000	0.00000	501	13062+	344	2310+	718+	23466+
742	Python	0.0	00:00:10	1	0	15	5268K	0B	0B	47742	1	sleeping	*0[1]	0.00000	0.00000	0	2254	242	60	24	2640
685	netbiosd	0.0	00:00:03	4	4	33	2544K	0B	0B	47559	1	sleeping	*0[1]	0.00000	0.00000	0	3009	156	123	40	1096
563	FileVaultEsc	0.0	00:00:07	3	2	34	1196K	0B	0B	47563	1	sleeping	*0[1]	0.00000	0.00000	0	2803	143	147	458	1795
562	TCCProfileSe	0.0	00:00:01	2	1	23	964K	0B	0B	47562	1	sleeping	*0[1]	0.00000	0.00000	0	2631	141	64	28	422
561	Certificates	0.0	00:00:08	3	2	34	1284K	0B	0B	47561	1	sleeping	*0[1]	0.00000	0.00000	0	2853	151	147	474	1925
560	MDMService	0.0	00:00:07	3	2	34	1292K	0B	0B	47560	1	sleeping	*0[1]	0.00000	0.00000	0	3648	148	147	453	1775
397	ocspd	0.0	00:00:47	2	1	39	1860K	0B	0B	47197	1	sleeping	*0[1]	0.00000	0.00000	0	10421+	164	6463+	145	3103+
944	mdmclient	0.0	00:01:52	5	3	98	10M	0B	0B	46944	1	sleeping	*1[255]	0.00000	0.00000	0	14613	405	7909	1893	62298
497	mdworker_sha	0.0	00:00:25	4	1	51	5320K	0B	0B	46497	1	sleeping	*0[1]	0.00000	0.00000	89	11746	188	1371	745	2971
486	mdworker_sha	0.0	00:00:01	4	1	51	9020K	0B	0B	46496	1	sleeping	*0[1]	0.00000	0.00000	89	23435	189	1880	922	6022
484	mdworker_sha	0.0	00:00:21	4	1	51	13M	0B	0B	46481	1	sleeping	*0[1]	0.00000	0.00000	89	9558	188	846	557	1434
472	mdworker_sha	0.0	00:00:00	4	1	55	7888K	0B	0B	46472	1	sleeping	*0[1]	0.00000	0.00000	89	22001	191	1794	909	6055
471	mdworker_sha	0.0	00:00:05	4	1	55	8392K	0B	0B	46471	1	sleeping	*0[1]	0.00000	0.00000	89	22149	191	1894	942	6586
481	mdworker_sha	0.0	00:00:13	3	1	55	3584K	0B	0B	46401	1	sleeping	*0[1]	0.00000	0.00000	501	5200	199	1070	676	2187
480	mdworker_sha	0.0	00:00:17	3	1	55	3508K	0B	0B	46402	1	sleeping	*0[1]	0.00000	0.00000	501	5239	200	1254	744	2772
399	mdworker_sha	0.0	00:00:16	3	1	55	3564K	0B	0B	46399	1	sleeping	*0[1]	0.00000	0.00000	501	5235	199	1198	725	2570
398	mdworker_sha	0.0	00:00:29	3	1	58	3936K	0B	0B	46398	1	sleeping	*0[1]	0.00000	0.00000	501	8039	221	2051	1052	5192
397	mdworker_sha	0.0	00:00:20	3	1	57	3988K	0B	0B	46397	1	sleeping	*0[1]	0.00000	0.00000	501	8060	221	2086	1061	5357
396	mdworker_sha	0.0	00:00:30	3	1	56	4096K	0B	0B	46396	1	sleeping	*0[1]	0.00000	0.00000	501	8094	221	2153	1101	5490
395	mdworker_sha	0.0	00:00:28	3	1	56	3980K	0B	0B	46395	1	sleeping	*0[1]	0.00000	0.00000	501	8015	221	1988	1029	4872
394	mdworker_sha	0.0	00:00:33	3	1	58	15M	0B	0B	46394	1	sleeping	*0[1]	0.11232	0.00000	501	8697	187	2417	1198	5861
388	mdworker_sha	0.0	00:00:18	3	1	58	14M	0B	0B	46388	1	sleeping	*0[1]	0.00000	0.00000	501	8139	186	1367	764	3684
387	mdworker_sha	0.0	00:00:20	3	1	58	15M	0B	0B	46387	1	sleeping	*0[1]	0.00000	0.00000	501	8452	187	1472	803	3921
383	com.apple.iC	0.0	00:03:36	2	1	59	5556K	0B	0B	46383	1	sleeping	*0[82]	0.00000	0.00000	501	5289	205	4610	2708	4517

talk

程序代写代做 CS编程辅导

(if you've got your laptop here) how many processes are running on your machine right now?



how about on your phone?

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Process: definition

basically: a *running* program

includes the code (instructions)

- registers/flags
- memory (stack and heap)
- permissions/privileges
- other resources (e.g. global variables; open files & network connections, address space mappings)

程序代写代做 CS编程辅导



program, and the current state/context:

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

processes as far as the eye can see

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



exact definition of process

Assignment Project Exam Help
depends on the OS
Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



WeChat: cstutorcs

so how do we manage them?
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

1 CPU per control-flow

程序代写代做 CS编程辅导

specific configurations only, e.g.

- distributed microcontrollers
- physical process control systems



WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
<https://tutorcs.com>

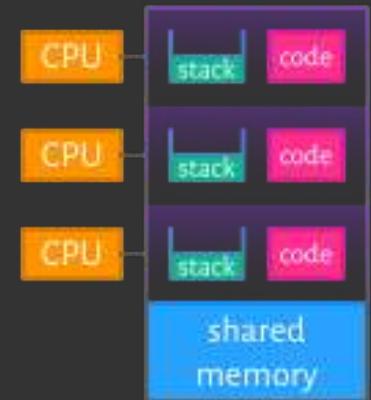
1 cpu per task, connected via a bus system

- Process management (scheduling) not required
- Shared memory access need to be coordinated

address
space 1

...

address
space n



1 CPU for all control-flows 程序代写代做 CS编程辅导

the OS may “emulate” one CPU for each control-flow



this is a **multi-tasking operating**

- support for memory protection essential
- process management (scheduling) required
- shared memory access need to be coordinated

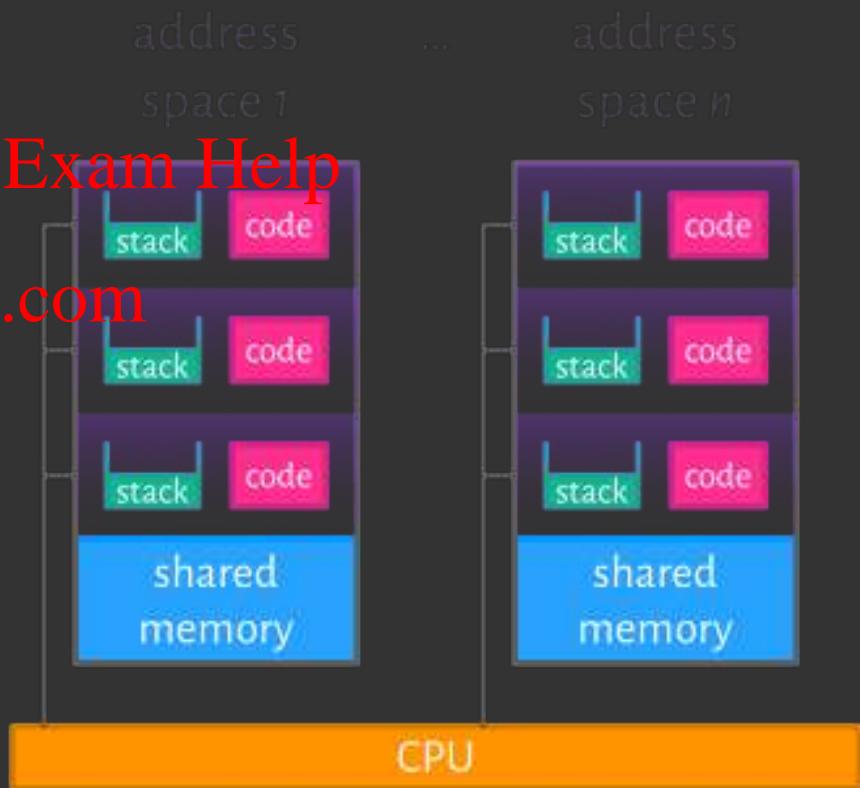
WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



Symmetric multiprocessing (SMP) 程序代写代做 CS编程辅导

all CPUs share the same physical address space (and have access to the same resources) so any process can be executed on any CPU



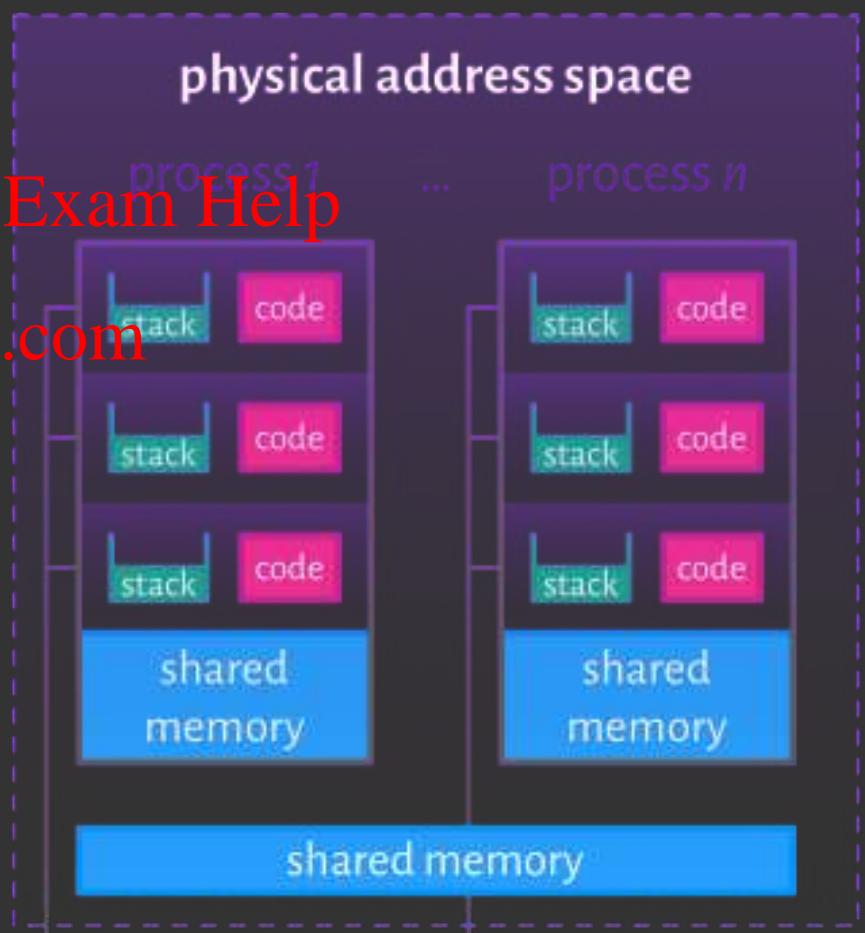
WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



Processes vs threads

程序代写代做 CS编程辅导

processes (as discussed **earlier**)



own registers, stack, resources, etc.

threads have their own registers

but share the other process resources

one process can create/manage many threads

WeChat: cstutorcs



Torvalds vs Threads

Linus Torvalds (torvalds@cs.helsinki.fi)
Tue, 6 Aug 1996 12:47:31 +0300 (EEST DST)

- Messages sorted by: [date] [thread] [subject] [author]
- Next message: [Ivan P. Ziller](#): "Re: Cops in net hash table"
- Previous message: [Linus Torvalds](#): "Re: [4.1] request ordering"

On Mon, 5 Aug 1996, Peter P. Fenwick wrote:
>
> We need to keep a clear no concept of threads. Too many people
> seem to confuse a thread with a process. The following discussion
> does not reflect the current state of Linux, but rather is an
> attempt to stay at a high level discussion.

NU!

There is NO reason to think that "threads" and "processes" are separate entities. That's how it's traditionally done, but I personally think it's a major mistake to think that way. The only reason to think that way is historical baggage.

Both threads and processes are really just one thing: a "context of execution". Trying to artificially distinguish different cases is just self-limiting.

A "context of execution", hereby called COE, is just the conglomerate of all the state of the COE. That state includes things like CPU state (registers etc), MMU state (page mappings), permission state (uid, gid) and various "communication states" (open files, signal handlers etc).

Traditionally, the difference between a "thread" and a "process" has been mainly that a threads has CPU state (+ possibly some other minimal state), while all the other context comes from the process. However, that's just one way of dividing up the total state of the COE, and there is nothing that says that it's the right way to do it. Limiting yourself to that kind of image is just plain stupid.

The way Linux thinks about this (and the way I want things to work) is that there is no such thing as a "process" or a "thread". There is only the totality of the COE (called "task" by Linux). Different COE's can share parts of their context with each other, and one subset of that sharing is the traditional "thread"/"process" swap, but that should really be seen as ONLY a subset (it's an important subset, but that importance comes not from design, but from standards: we obviously want to run standard-conforming threads programs on top of Linux too).

In short: do NOT design around the thread/process way of thinking. The kernel should be designed around the COE way of thinking, and then the pthreads library can expect the limited pthreads interface to users who want to use that way of looking at COEs.

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

"Depends on the implementation" ...

程序代写代做 CS 编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Process Control Blocks (PCBs)

- process ID
- process state: {created, ready, executing, blocked, suspended, bored ...}
- scheduling attributes: priorities, deadlines
- CPU state: (e.g. registers, stack pointer)
- memory attributes/privileges: permissions, limits, shared areas
- allocated resources: open/requested devices and files, etc.



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



process control
block(s)

程序代写代做 CS编程辅导



WeChat: cstutorcs

a data structure for processes
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Process states

程序代写代做 CS编程辅导

- **created**: the task is ready to run, but has not yet been considered by any dispatcher
- **ready**: ready to run (waiting for the dispatcher)



- **running**: holds a CPU and executes
- **blocked**: not ready to run (waiting for a resource)
- **suspended**: swapped out of main memory (e.g. waiting for main memory space)

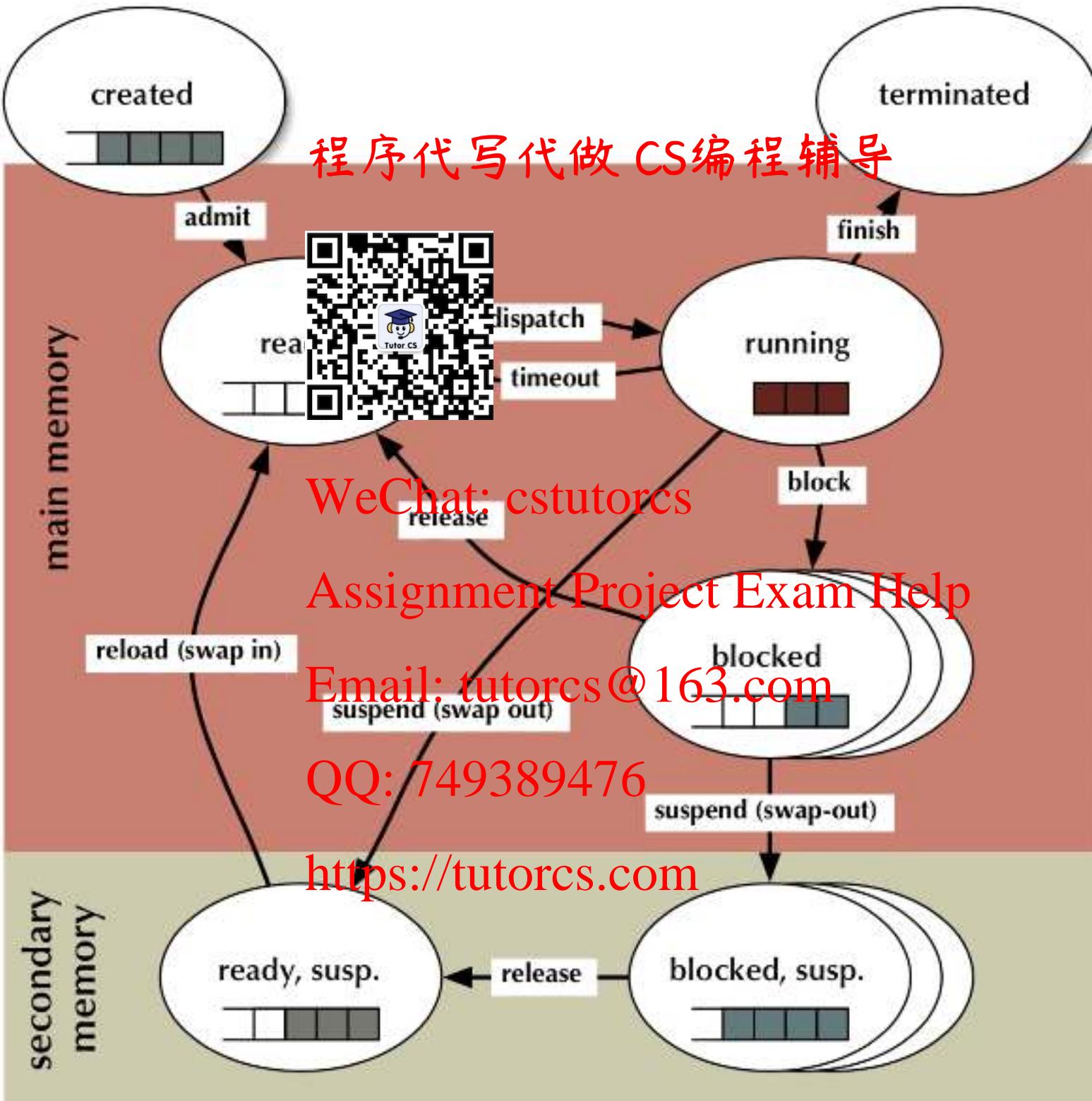
Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

WeChat: cstutorcs

Assignment Project Exam Help



First come, first served (FCFS) scheduling

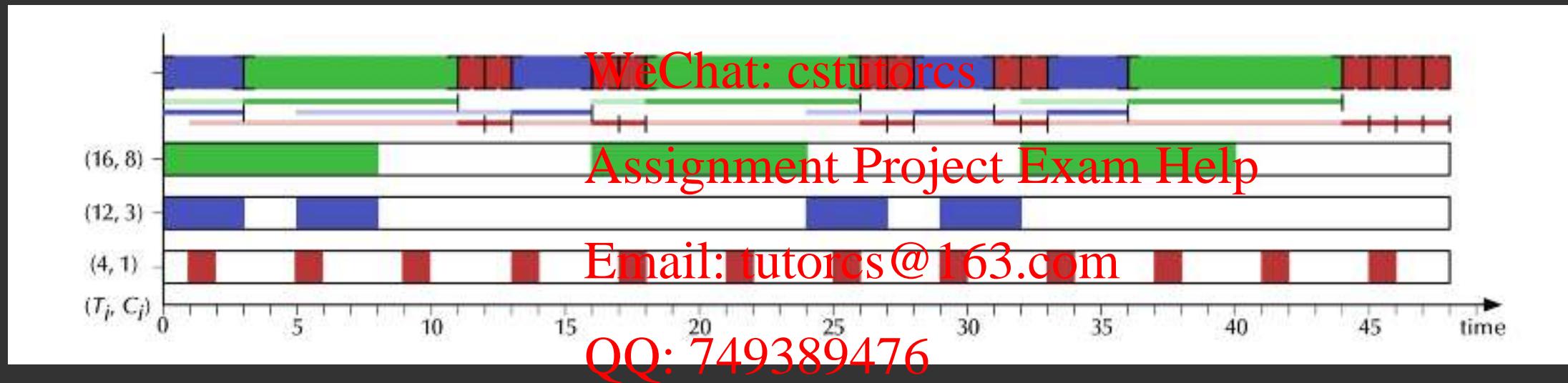


<https://tutorcs.com>

- **Waiting time:** 0..11, average: 5.9
- **Turnaround time:** 3..12, average: 8.4

FCFS (again)

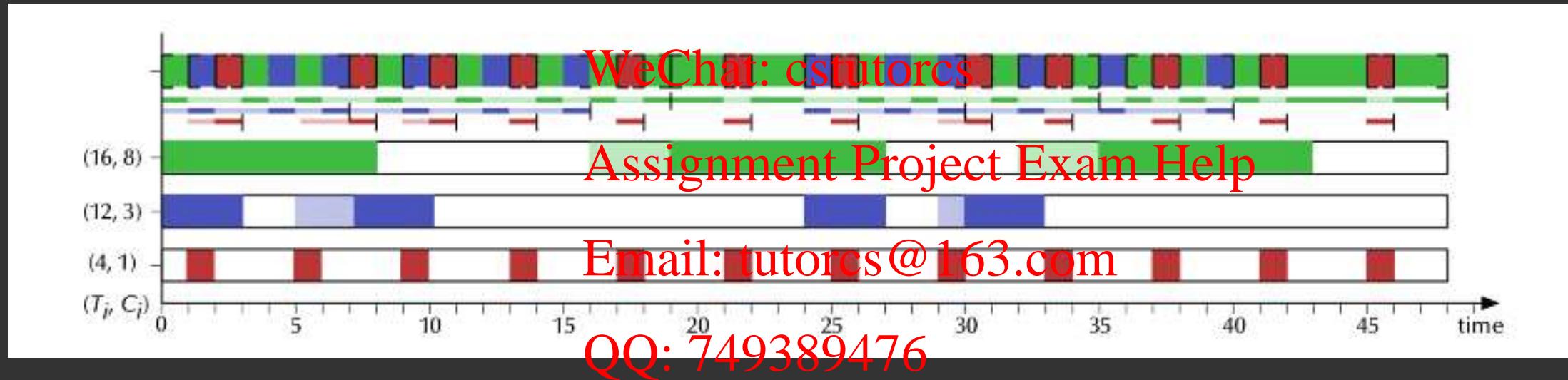
程序代写代做 CS 编程辅导



<https://tutorcs.com>

- **Waiting time:** 0..11, average: 5.4 (was 5.9 before)
- **Turnaround time:** 3..12, average: 8.0 (was 8.4 before)

Round-robin (RR) scheduling



- Waiting time: 0..5, average: 1.2
- Turnaround time: 1..20, average: 5.8

<https://tutorcs.com>

optimised for swift initial responses, but “stretches out” long tasks

talk

程序代写代做 CS编程辅导

when might you want to use FCF? what about LRU? how about RR?



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



Again, a whirlwind tour of OSes

程序代写代做 CS 编程辅导

remember the concepts



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



WeChat: cstutorcs

go build your own in labs.
Assignment Project Exam Help.

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Questions

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Fun With Operating Systems



- Kernel writing 101
- Linux on an 8bit AVR
- How to make an operating system (WikiHow)

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Further Reading

程序代写代做 CS 编程辅导

Essentials of Computer Organis



Architecture - Chapter 8.2: Operating Systems

Nick Moffitt's \$7 History of Unix

WeChat: cstutorcs

LGR Tech Tales - How Digital Research Almost Ruled PCs
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>