زمان‌بندی (رایانش)

زمان‌بندی[۱] (به انگلیسی: scheduling) در رایانش، روش انتساب کار به منابعی است که کار را انجام می دهند. در اینجا «کار» می تواند عناصر رایانشی مشهود باشد، مثل ریسه، فرایند، یا جریان‌داده باشد که به نوبه خود به منابع سخت‌افزاری مثل پردازنده، اتصال شبکه، یا کارت توسعه زمان‌بندی می‌گردد.

مولفه **زمان‌بندی** پردازش‌ها (به انگلیسی: Process Scheduler) در سیستم‌عامل بخشی از سیستم‌عامل است که تصمیم می‌گیرد که کدام پردازش چه زمانی و به چه مدتی اجرا شود.[۲]

ایده اصلی زمان‌بندی این است که برای استفاده بهینه از زمان پردازنده، با این فرض که پردازش‌های قابل اجرا وجود دارند، یک پردازش همواره باید در حال اجرا باشد. اگر تعداد پردازش‌ها بیش از تعداد پردازنده‌ها باشد، در هر لحظه برخی از پردازش‌ها در حال اجرا نخواهند بود. این پردازش‌ها منتظر اجرا هستند. تصمیم‌گیری اینکه با داشتن مجموعه‌ای از پردازش‌های قابل اجرا، کدام پردازش در مرحله بعد اجرا شود، تصمیم اصلی است که یک زمان‌بند باید بگیرد.[۲]

انواع زمان‌بندها

سه نوع زمان‌بند در سیستم‌عامل‌ها وجود دارد:

- زمان‌بندی بلندمدت یا زمان‌بند کار یا زمان‌بند پذیرش - که درجه چند برنامگی را مشخص می‌کند
- زمان‌بند میان‌مدت - که پروسه‌ها را بین دیسک و حافظه مبادله می‌کند.
- زمان‌بند کوتاه‌مدت - که پردازنده را به فرایندها اختصاص می‌دهد.

بسته به نوع سیستم‌عامل، ممکن است از همه این زمان‌بندها استفاده نشود. برای مثال در سیستم‌عامل‌های اشتراک زمانی معمولاً از زمان‌بند بلند مدت استفاده نمی‌شود.

زمان‌بند بلندمدت

این زمان‌بند تصمیم می‌گیرد که کدام کارها یا فرایندها می‌توانند در صف آماده قرار گیرند. این زمان‌بند کارها را از روی دیسک برداشته و به حافظه اصلی برده و در صف آماده قرار می‌دهد تا پردازنده به آن‌ها اختصاص یابد. وقتی که تلاشی برای اجرای یک برنامه انجام می‌شود، زمان‌بند کار یا آن برنامه را می‌پذیرد و به حافظه اصلی می‌برد یا اجرای آن را به تأخیر می‌اندازد. این زمان‌بند درجه چند برنامگی را تعیین می‌کند. منظور از درجه چند برنامگی این است که همواره چه تعداد فرایندی در حافظه اصلی وجود داشته باشند تا پردازنده به آن‌ها اختصاص یابد.

زمان‌بند میان‌مدت

این زمان‌بند، کارها را به‌طور موقت از روی حافظه اصلی برمی‌دارد و در حافظه جانبی (مثل دیسک) قرار می‌دهد (یا برعکس). این تکنیک بیشتر با نام مبادله یا صفحه‌بندی شناخته می‌شود. زمان‌بند میان‌مدت ممکن است تصمیم بگیرد که پروسه‌ای که برای مدتی غیرفعال بوده، یا پروسه‌ای که اولویت پایینی دارد، یا پروسه‌ای که مکرراً نقص صفحه تولید می‌کند، یا پروسه‌ای که مقدار زیادی از حافظه را اشغال کرده را از حافظه اصلی خارج کرده و موقتاً در دیسک قرار دهد. سپس وقتی که حافظه بیشتری در دسترس قرار گرفت یا پروسه دیگر منتظر منابع نبود، آن را از دیسک برداشته و در حافظه قرار می‌دهد.

زمان‌بند کوتاه‌مدت

زمان‌بند کوتاه‌مدت یا *زمان‌بند پردازنده*، یکی از پروسه‌هایی که در صف آماده قرار دارد را انتخاب کرده و پردازنده را به آن پروسه اختصاص می‌دهد تا اجرا شود. این زمان‌بند بعد از رخ دادن یک وقفه ساعت، یک وقفه ورودی/خروجی، اجرای یک فراخوان سیستمی، یا دریافت یک سیگنال فعال می‌شود و یکی از پروسه‌های منتظر در صف اجرا را برای پردازش شدن انتخاب می‌کند؛ بنابراین زمان‌بند کوتاه‌مدت نسبت به زمان‌بندهای بلندمدت و میان‌مدت وقت کمتری برای تصمیم‌گیری دارد و ممکن است در ثانیه چند بار اجرا شود. زمان‌بند کوتاه‌مدت می‌تواند انحصاری یا غیر-انحصاری باشد. اگر به صورت غیر انحصاری باشد، می‌تواند پردازنده را از یک پروسه در حال اجرا بگیرد و آن را به پروسه دیگری اختصاص دهد. زمان‌بند انحصاری این کار را انجام نمی‌دهد و پروسه خودش باید پردازنده را رها کند. (به این حالت چند برنامگی تعاونی می‌گویند)

اعزام‌کننده

اعزام‌کننده یا (به انگلیسی: Dispatcher) یکی دیگر از مؤلفه‌هایی است که در زمان‌بندی پردازنده درگیر هستند.
اعزام‌کننده ماژولی است که کنترل پردازنده را به فرایندی می‌دهد که زمانبند کوتاه‌مدت آن را انتخاب کرده‌است. این کار از سه مرحله تشکیل می‌شود:

- تعویض زمینه (ذخیره ثبات‌ها و PCB فرایند فعلی و بارگذاری ثباتها و PCB فرایند جدید)
- تعویض به مد کاربری

- پرش به مکان مناسب از برنامه کاربر که قرار است پردازش از آنجا ادامه یابد.

اعزام‌کننده، مقادیر شمارنده برنامه را بررسی کرده و دستورالعمل‌ها را واکشی می‌کند و داده‌ها را در ثبات‌ها بارگذاری می‌منند. اعزام‌کننده باید بسیار سریع عمل کند. زیرا هر بار که قرار است پروسه‌ها تعویض شوند، این برنامه باید فراخوانی شود. در هنگام تعویض زمینه، پردازنده برای کسری از زمان بی‌کار می‌شود؛ بنابراین از تعویض زمینه‌های غیرضروری باید پرهیز کرد. مدت زمانی که طول می‌کشد تا اعزام‌کننده یک فرایند را متوقف کرده و فرایند جدیدی را آغاز کند، تحت عنوان **تأخیر اعزام‌کننده** شناخته می‌شود.

سیستم‌عامل‌های چندوظیفه‌ای

سیستم‌عامل چندوظیفه‌ای سیستم‌عاملی است که در آن بیش از یک پردازش بتواند هم‌زمان در حال اجرا باشد.

سیستم‌عامل‌های چندوظیفه‌ای به‌طور عمده به دو قسم تقسیم می‌شوند:[۲]

چندوظیفه‌ای پیشگیرانه (قبضه کردنی یا غیر انحصاری)

چندوظیفه‌ای پیشگیرانه (به انگلیسی: preemptive multitasking): در این سیستم‌عامل‌ها، مؤلفه زمان‌بندی تصمیم می‌گیرد که چه زمانی یک پردازش متوقف و پردازش بعدی شروع به اجرا شود. مدت زمانی که یک پردازش پس از گذر آن متوقف می‌شود معمولاً از پیش تعیین‌شده‌است، و آن را برش‌زمانی[پ ۱] پردازش می‌نامند. در بسیاری از سیستم‌های عامل مدرن، مقدار برش‌زمانی به صورت پویا و تابعی از رفتار پردازش و سیاست سیستم محاسبه می‌شود.

چندوظیفه‌ای مشارکتی (انحصاری)

چندوظیفه‌ای مشارکتی (به انگلیسی: cooperative multitasking): در این سیستم‌ها پردازش تا وقتی که به صورت داوطلبانه اجرا را متوقف نسازد، در حال اجراست و سیستم‌عامل دخالتی نمی‌کند. دو نمونه از این سیستم‌عامل‌ها، ویندوز ۳٫۱ و مک اواس ۹ است.

الگوریتم‌های زمان‌بندی پردازش

زمان‌بندی پردازش‌گر با مسئله تصمیم‌گیری انتخاب پردازش بعدی برای اجرا سروکار دارد. برای این کار الگوریتم‌های مختلفی وجود دارد. در این بخش به شرح برخی از آن‌ها می‌پردازیم.[۳]

زمان‌بندی اجرا به ترتیب ورود

الگوریتم اجرا به ترتیب ورود (به انگلیسی: **F**irst **C**ome **F**irst **S**erved)، ساده‌ترین الگوریتم زمان‌بندی و یک الگوریتم انحصاری است. در این روش، پروسه‌ها به ترتیبی که وارد می‌شوند، در صف آماده قرار می‌گیرند و پردازشگر به ترتیب آن‌ها

را پردازش می‌کند. پیاده‌سازی این الگوریتم به آسانی توسط یک صف قابل انجام است. هنگامی که پردازشگر آزاد است، آن را به پردازشی که در سر صف قرار دارد تخصیص می‌دهیم، و این پردازش را از صف حذف می‌کنیم.

- از آنجا که تعویض زمینه تنها در هنگام خاتمه فرایند صورت می‌گیرد، و همچنین نیازی به سازماندهی مجدد صف نیست، سربار زمانبندی اندک است.
- توان عملیاتی این روش ممکن است پاییم باشد، زیرا ممکن است یک فرایند برای یک مدت طولانی پردازنده را در اختیار داشته باشد و آن را رها نکند.
- زمان برگشت، زمان انتظار و زمان پاسخ هم می‌تواند بالا باشد. (بنا به دلیل بالا)
- هیچ گونه اولویت‌بندی وجود ندارد.
- این روش مبتنی بر صف است.

از این الگوریتم به ندرت استفاده می‌شود یا به عنوان الگوریتم ثانویه و به همراه دیگر الگوریتم‌ها استفاده می‌شود.

زمان‌بندی نخست کوتاه‌ترین کار

الگوریتم **نخست کوتاه‌ترین کار** (به انگلیسی: **S**hortest **J**ob **F**irst) یک مدل از الگوریتم های انحصاری است. پردازنده به پروسه‌ای اختصاص می‌یابد که طول اجرای آن از همه کمتر است. هنگامی که پردازش‌گر آماده اجرا است، پردازشی با کوتاه‌ترین طول اجرای بعدی انتخاب می‌شود. این الگوریتم قابل پیاده‌سازی نیست. چرا که باید طول اجرای هر پروسه را محاسبه کرد که در عمل این کار امکان‌پذیر نیست. طول اجرای پروسه یا باید محاسبه شود (که کاری وقت‌گیر و بیهوده است) یا باید تخمین زده شود. به همین دلیل در مواقعی استفاده می‌شود که طول اجرای پروسه از قبل مشخص باشد. می‌توان اثبات کرد که این الگوریتم بهینه است، و به ازای هر مجموعه از پردازش‌ها کم‌ترین زمان پردازش متوسط را ارائه می‌دهد. این الگوریتم دارای مشکل گرسنگی است. به این معنی که اگر به صورت مداوم فرایندهایی به طول اجرای پایین وارد سیستم شوند، ممکن است پروسه‌هایی با طول اجرای طولانی‌تر هرگز اجرا نشوند. مشکل اساسی این الگوریتم سخت بودن طول اجرای بعدی هر کدام از پردازش‌ها است.

زمانبندی کوتاهترین زمان باقی‌مانده

الگوریتم کوتاهترین زمان باقی‌مانده (به انگلیسی: **S**hortest **R**emaining **T**ime) مشابه الگوریتم «نخست کوتاه‌ترین کار» و یک الگوریتم غیر انحصاری است که مناسب برای سیستم های اشتراک زمانی است . در این الگوریتم، زمانبند سیستم پروسه‌ای را اجرا می‌کند که نیاز به کمترین زمان برای تکمیل شدن دارد. در این الگوریتم هم باید طول زمان باقی‌مانده برای تکمیل پروسه تخمین زده شود یا از قبل مشخص باشد؛ بنابراین این الگوریتم هم قابل پیاده‌سازی نیست.

- اگر در هنگام اجرای یک پروسه، پروسه کوچکتری وارد سیستم شود، پردازنده به پروسه جدید داده می‌شود و اجرای پروسه بزرگتر به دو بلاک محاسباتی مجزا تقسیم می‌شود. این کار باعث ایجاد سربار اضافی از طریف تعویض زمینه می‌شوند. همچنین زمانبند باید هر پروسه جدید را در مکان خاصی از صف آماده قرار دهد که این کار هم یک سربار اضافی دیگر است. چرا که حذف و اضافه کردن در صف کار سریعی نیست.

- از این روش دیگر استفاده نمی‌شود.
- برای استفاده از این روش ما حداقل به دو پروسه با اولویت‌های مختلف نیاز داریم
- این الگوریتم دارای مشکل گرسنگی است. چرا که ممکن است به‌طور مداوم پروسه‌های کوتاه وارد سیستم شوند و اجرای پروسه‌های طولانی به تأخیر بیفتد.

بالاترین نسبت پاسخ

سپس بالاترین نسبت پاسخ (به انگلیسی: **H**ighest **R**esponse **R**atio **N**ext) یک الگوریتم از نوع انحصاری است و مشابه الگوریتم «نخست کوتاه‌ترین کار» است که مشکل گرسنگی فرآیند ها را برطرف کرده است. در این الگوریتم، اولویت هر فرآیند، هم به مدت زمان اجرای آن و هم به مدت زمانی که در صف آماده منتظر دریافت پردازنده بوده، بستگی دارد. هر چه یک فرایند بیشتر در صف آماده منتظر دریافت پردازنده بماند، اولویتش بالاتر خواهد رفت. به این ترتیب این الگوریتم پدیده گرسنگی را برطرف می‌کند و کارهای طولانی مدت هم بالاخره اجرا خواهد شد. در این الگوریتم، اولویت هر فرایند به صورت زیر تعیین می‌شود:

$$Priority = \frac{waiting\ time + estimated\ run\ time}{estimated\ run\ time} = 1 + \frac{waiting\ time}{estimated\ run\ time}$$

زمان‌بندی اولویت

الگوریتم **زمان‌بندی اولویت** (به انگلیسی: Priority scheduling), یک مقدار اولویت به هر کدام از فرآیندها تخصیص داده می‌شود، و فرآیند با اولویت بالاتر برای اجرای بعدی انتخاب می‌شود. فرآیند های با اولویت یکسان به صورت اجرا به ترتیب ورود زمان‌بندی می‌شوند.[۳]

یکی از معایب این الگوریتم , گرسنگی فرآیند های با اولویت پایین تر است که ممکن است هیچ گاه اجرا نشوند.

زمان‌بندی نوبت گردشی

الگوریتم زمان‌بندی نوبت گردشی (به انگلیسی: Rond-Robin) یک الگوریتم غیرانحصاری است و می‌توان آن را پیاده‌سازی کرد. در این الگوریتم، زمانبند به هر پروسه یک واحد زمانی ثابت اختصاص می‌دهد و سپس در بین آن‌ها گردش می‌کند. به عبارتی دیگر پردازنده هر فرایند را برای مدت زمان کوتاهی اجرا کرده و سپس به سراغ فرایند بعدی می‌رود. برش زمانی معمولاً بین ۱۰ تا ۱۰۰ میلی‌ثانیه است و پردازنده هر پروسه را به این میزان اجرا می‌کند. برش زمانی نباید کمتر از زمان تعویض زمینه باشد. برش زمانی توسط یک وقفه ساعت اتفاق می‌افتد.

- این الگوریتم سربار اضافه به سیستم تحمیل می‌کند. مخصوصاً اگر برش رمانی کوتاه باشد.
- گرسنگی اتفاق نمی‌افتد، زیرا هیچ اولویتی بین پروسه‌ها وجود ندارد.
- زمان پاسخگویی متوسط است، زمان انتظار به تعداد پروسه‌ها بستگی دارد.

- به خاطر زمان انتظار بالا، بن‌بست معمولاً اتفاق نمی‌افتد.

از این الگوریتم در سیستم‌های اشتراک زمانی هم استفاده می‌شود.

صف چندگانه فیدبک

صف چندگانه فیدبک (به انگلیسی: Multilevel queue scheduling) وقتی استفاده می‌شود که بتوان پروسه‌ها را به آسانی به گروه‌هایی دسته‌بندی کرد. برای مثال یک روش رایج برای تقسیم کردن این است که پروسه‌های پیش‌زمینه (به انگلیسی: foreground) (که تعاملی هستند) در یک گروه و پروسه‌های پس‌زمینه (به انگلیسی: background) (که دسته‌ای هستند) در گروهی دیگر قرار گیرند. این دو گروه، احتیاج به زمان پاسخ‌دهی متفاوتی دارند. هر پروسه وارد یک صف خاص می‌شود. صف‌ها نسبت به هم اولویت دارند. مثلاً اولویت اجرای پروسه‌های پیش‌زمینه از پروسه‌های پس زمینه بیشتر است. هر صف می‌تواند الگوریتم زمان‌بندی مخصوص به خود را داشته باشد. مثلاً یک صف از زمانبندی نوبت گردشی و صف دیگر از زمانبندی SRT استفاده کند. این الگوریتم در سیستم‌عامل‌های مدرن هم استفاده می‌شود.

خلاصه

الگوریتم زمان‌بندی	پردازنده سربار	توان عملیاتی	زمان برگشت کار	زمان پاسخ‌دهی
FCFS	پایین	پایین	بالا	بالا
نخست کوتاه‌ترین کار	متوسط	بالا	متوسط	متوسط
کوتاه‌ترین زمان باقی‌مانده	متوسط	بالا	متوسط	متوسط
بالاترین نسبت پاسخ	متوسط	بالا	متوسط	متوسط
زمان‌بندی اولویت	متوسط	پایین	بالا	بالا
زمان‌بندی نوبت گردشی	بالا	متوسط	متوسط	بالا
صف چندسطحی فیدبک	بالا	بالا	متوسط	متوسط
دستی	پایین	متوسط	بالا	بالا

زمان‌بندی در سیستم‌عامل‌های مختلف

مایکروسافت

داس و ویندوزهای اولیه

- مایکروسافت داس و ویندوز ابتدایی مایکروسافت قابلیت چند پردازشی نداشتند لذا به زمان‌بندی نیازی نبود.

- ویندوز ۳٫۱ از زمان‌بندی ناپیشگیرانه[پ ۲] یا مشارکتی[پ ۳] استفاده می‌کرد به این معنا که یک پردازش خاص متوقف نمی‌شد تا زمانی که به پایان برسد یا به سیستم عامل بگوید که دیگر نیازی به پردازنده ندارد در نتیجه پردازنده به پردازش دیگری اختصاص می‌یابد.
- ویندوز ۹۵ یک زمان‌بندی پیشگیرانه ناقص و ابتدایی را ارائه کرد؛ البته به منظور پشتیبانی از پردازش‌های ۱۶ بیتی از همان روش ناپیشگیرانه برای این پردازش‌ها استفاده شد.

ویندوز ان‌تی

سیستم‌عامل‌های مبتنی بر ویندوز ان‌تی از زمان‌بندی بازخوردی استفاده می‌کنند. ۳۲ سطح اولویت تعریف شده‌است، از ۰ تا ۳۱. سطوح ۰ تا ۱۵ اولویت معمولی و سطوح ۱۶ تا ۳۱ اولویت بی‌درنگ را دارا می‌باشند؛ اولویت ۰ برای سیستم عامل محفوظ است.
هسته سیستم‌عامل می‌تواند سطح اولویت یک ریسه[پ ۴] را بر اساس نیاز آن به ورودی/خروجی و پردازنده یا تعاملی بودن آن تغییر دهد؛ بدین صورت که سطح اولویت ریسمان‌های در تنگنای ورودی/خروجی[پ ۵] و ریسمان‌های مربوط به پردازش‌های محاوره‌ای را بالا می‌برد تا میزان پاسخگویی آن‌ها افزایش یابد، از طرفی سطح اولویت ریسه‌های در تنگنای پردازشگر[پ ۶] را کاهش می‌دهد.

ویندوز سرور ۲۰۰۸ و ویندوز ویستا

در ویندوز ویستا زمان‌بندی به گونه‌ای تغییر کرد که به جای استفاده از وقفه‌های زمانی[پ ۷] برای تعیین مدت اجرای یک ریسه، از ثبات شمارندهٔ سیکل[پ ۸] موجود در پردازنده‌های مدرن استفاده شود.

ویندوز ویستا همچنین از زمان‌بندی اولویت در صف ورودی/خروجی استفاده می‌کند تا مانع از تداخل پردازش‌های کم اولویتی همچون یکپارچه‌سازی دیسک سخت با پردازش‌های در حال اجرا شود.

لینوکس

در سیستم‌عامل لینوکس هر پردازشی در یکی از رده‌های سیاست زمان‌بندی قرار می‌گیرد، که معمولاً یکی از مقادیر زیر است:[۴]

- SCHED_OTHER: سیاست زمان‌بندی برای پردازش‌های عادی
- SCHED_FIFO: سیاست زمان‌بندی خروج به ترتیب ورود برای پردازش‌های بی‌درنگ
- SCHED_RR: سیاست چرخشی برای پردازش‌های بی‌درنگ

هر کدام از رده‌های سیاست زمان‌بندی الگوریتم خود را برای انتخاب پردازش بعدی دارند. رده‌های بی‌درنگ اولیت بیشتری نسبت به اولویت SCHED_OTHER دارند. اگر در این رده‌ها پردازشی برای اجرا وجود داشته باشد، الگوریتم زمان‌بند رده SCHED_OTHER اجرا نمی‌شود.

در رده‌های SCHED_FIFO و SCHED_RR زمان‌بندی با توجه به مقدار اولیت بلادرنگی[پ ۹] زمان‌بندی می‌شوند که مقداری بین ۱ تا ۹۹ دارد. مقدار بالاتر متناظر با اولویت بالاتر است.[۲] در این رده، تا زمانی که پردازشی با اولیت بالاتر وجود داشته باشد، سایر پردازش‌ها فرصت اجرا پیدا نمی‌کنند.

در رده SCHED_OTHER پردازش‌ها با توجه به مقدار اولویت nice که مقداری بین ۲۰- تا ۱۹+ است زمان‌بندی می‌شوند. مقدار بالاتر متناظر با اولیت کمتر است.[۲]

برای زمان‌بندی پردازش‌های رده SCHED_OTHER از نسخه ۲٫۶٫۲۳ به بعد هسته لینوکس از «زمان‌بند کاملاً عادلانه»[پ ۱۰] یا CFS استفاده می‌شود.[۵] پیش از این زمان‌بند، زمان‌بند مورد استفاده در هسته لینوکس نسخه ۲٫۶، زمان‌بند مرتبه ۱ یا زمان‌بند O(1) بود.[۶]

در نسخه‌های هسته لینوکس ۲٫۴ تا قبل از ۲٫۶، الگوریتم زمان‌بندی نسبتاً ساده بود، و زمان‌بند هر یک از پردازش‌ها را بررسی می‌کرد و به آن امتیازی می‌داد، و پردازشی که بیشترین امتیاز را به دست می‌آورد را برای اجرا انتخاب می‌کرد.[۷] بنابراین پیچیدگی زمانی این الگوریتم از مرتبه O(N) بود. با اینکه الگوریتم نسبتاً ساده بود، ولی نسبتاً ناکارامد بود و برای سامانه‌های بی‌درنگ مناسب نبود.

فری‌بی‌اس‌دی

فری‌بی‌اس‌دی از روش صف چندگانه فیدبک با اولویتهایی بین ۰ تا ۲۵۵ استفاده می‌کند. اولویت‌های ۰ تا ۶۳ برای وقفه‌ها رزرو شده‌اند. ۶۴ تا ۱۲۷ برای نیمه بالایی هسته، ۱۲۸ تا ۱۵۹ برای ریسه‌های بی‌درنگ کاربر، ۱۶۰ تا ۲۲۳ برای ریسه‌های اشتراک زمانی کاربر، ۲۲۴ تا ۲۵۵ هم برای ریسه‌های بیکار کاربر رزرو شده‌اند. مشابه لینوکس، فری‌بی‌اس‌دی هم یک صف فعال دارد، اما فری‌بی‌اس‌دی علاوه بر این صف، یک صف بیکار هم دارد.

پانویس

1. *timeslice*
2. *non-preemptive*
3. *cooperative*
4. *thread*
5. *I/O bound*
6. *CPU-bound*
7. *interval-time interrupt*
8. *cycle counter register*
9. *realtime priority*
10. *Completely Fair Scheduler*

منابع

1. «زمان‌بندی» [مدیریت-مدیریت پروژه] هم‌ارزِ «scheduling»، منبع: گروه واژه‌گزینی (https://apll.ir/%D9%88%D8%A7%DA%98%D9%87%E2%80%8C%DA%AF%D8%B2%DB%8C%D9%86%DB%8C/) . جواد میرشکاری، ویراستار. دفتر یازدهم. فرهنگ واژه‌های مصوب فرهنگستان. تهران: انتشارات فرهنگستان زبان و ادب فارسی. شابک ۳-۴۵-۶۱۴۳-۶۰۰-۹۷۸ (ذیل سرواژهٔ زمان‌بندی 1)
2. کتاب Linux Kernel Development، ویرایش سوم، نوشته رابرت لاو، فصل ۴
3. کتاب مفاهیم سیستم عامل، ویرایش هشتم، نوشته آبراهام سیلبرشاتز
4. صفحات راهنمای لینوکس - تابع sched_setscheduler (http://www.kernel.org/doc/man-pages/online/pages/man2/sched_setscheduler.2.html)
5. مستندات لینوکس - سند طراحی CFS (http://www.kernel.org/doc/Documentation/scheduler/sched-design-CFS.txt)
6. دولوپرورکز آی‌بی‌ام - نگاهی به درون زمان‌بند لینوکس (http://www.ibm.com/developerworks/linux/library/l-scheduler/index.html)
7. نگاهی به درون زمان‌بند کاملاً عادلانه لینوکس ۲٫۶ (http://www.ibm.com/developerworks/linux/library/l-completely-fair-scheduler/)

- **ویکی‌پدیای انگلیسی**

آخرین ویرایش ۱ سال پیش **توسط Mo.hajesmaili انجام شده**

ویکی‌پدیا