همچنین وظایف آن در بخش های قبلی تشریح شد، این مطلب را به وضوح تایید می کند.

یک روش برای رفع مشکلات و چالش های مذکور، انتقال وظیفه ایجاد و مدیریت نخ ها از برنامه نویسان به کامپایلرها و کتابخانه هاست. به نخ هایی که توسط این استراتژی تولید می شوند اصطلاحاً نخ های ضمنی (Implicit Thread) اطلاق می شود.

در چنین استراتژی، توسعه دهندگان (برنامه نویسان) تنها نکاتی که می توانند موازی اجرا شوند را شناسایی می کنند و کتابخانه ها و کامپایلرهای مربوطه در نظر گرفته شده، جزییات لازم برای ایجاد و مدیریت نخ ها را تعیین می کنند.

(۲۱) استخر نخ (Thread Pool)

مثال وب سرور چند نخی که در بخش های قبلی بیان گردید را در نظر بگیرید. در این مثال وب سرور به محض دریافت یک درخواست جدید، یک نخ جدید برای سرویس دهی به این درخواست ایجاد و سپس آن را اجرا می کرد.

در چنین سناریویی، در حالیکه ایجاد یک نخ جداگانه برای سرویس دهی به درخواست ها بهتر از ایجاد یک فرآیند جداگانه است، وب سرور چند نخی همچنان دارای مشکلات بالقوه ای می باشد:

- مساله اول مربوط به زمان ایجاد نخ است، همراه با این حقیقت که نخ ها پس از اتمام کار خود حذف می شوند.

- مساله دوم که حادتر است آن است که اگر برای هر سرویس دهی یک نخ ایجاد شود، آنگاه محدودیتی برای تعیین نخ های هم زمان فعال در سیستم وجود نخواهد داشت. بنابراین وجود نخ های فعال زیاد می تواند منابع سیستم نظیر CPU یا حافظه را بیش از حد

هدر دهد.

یک راه‌حل برای روبارویی با مشکلات فوق، استفاده از استخر نخ (Thread Pool) می‌باشد.

ایده‌ی استخر نخ بدین نحو است که به هنگام راه‌اندازی فرآیند، تعدادی نخ ایجاد و در یک استخر منتظر دریافت تسک می‌شوند. در این صورت، هنگامی که سرور درخواستی را دریافت می‌کند به جای ایجاد یک نخ، نخی را از استخر برداشته و سرویس‌دهی را به آن محول می‌کند. (در صورت وجود نخ در استخر)

با این حال اگر نخی در استخر موجود نباشد، تسک دریافتی تا آزاد شدن یک نخ، در صف انتظار باقی می‌ماند. هنگامی که نخ امتحان یافته به درخواست، اجرایش تمام شد به نخ‌های استخر پیوسته و منتظر دریافت تسک جدیدی می‌شود.

با توجه به مباحث مطرح شده در بالا، استخر نخ مزایای زیر را به همراه دارد:

۱- افزایش سرویس‌دهی به یک درخواست با نخ موجود بسریعتر از ایجاد یک نخ جدید است.

۲- استخر نخ، تعداد نخ‌های موجود را محدود می‌کند. این محدودیت بطور خاص برای سیستم‌هایی که نمی‌توانند تعداد زیادی از نخ‌های همروند را پشتیبانی کنند، اهمیت دارد.

۳- مجزا کردن تسکی که باید اجرا شود از مکانیزم ایجاد تسک، اجازه اجرای استراتژی‌های مختلفی را برای تسک فراهم می‌کند. بنابراین تسک‌های توانند به صورت دوره‌ای زمانبندی شوند.

نکته: تعداد نخ‌ها در استخر نخ را می‌توان به صورت اکتشافی (heuristic) بر اساس عواملی نظیر: تعداد CPU در سیستم کامپیوتری، مقدار حافظه فیزیکی

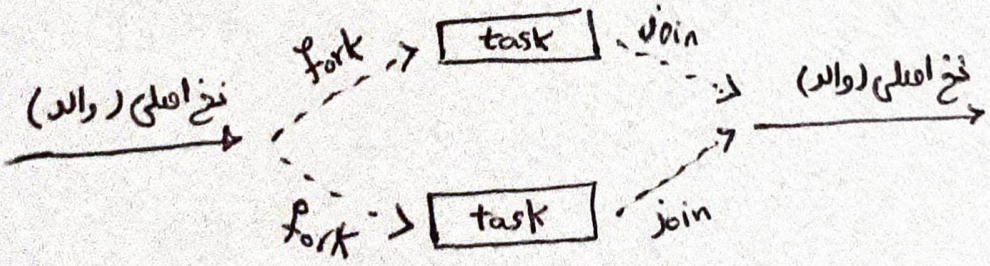
قابل دسترس در سیستم، تعداد درخواست های مورد انتظار مشتری و موارد مشابه تعیین نموده

می توان با در نظر گرفتن یک معماری پیچیده تر ساختار انعطاف پذیرتری را برای استخر نخ متصور شد که در آن تعداد نخ ها می تواند به صورت پویا و براساس الگوی استفاده تعیین شود. چنین معماری مزیت بیشتری را به همراه دارد و هنگامی که تعداد کارهای ارسالی کم می شود، حافظه کمتری مصرف می شود.

(۲۲) مفهوم موازی سازی Fork-Join

مدل Fork-Join یک استراتژی برای ایجاد نخ ها می باشد. در این مدل، نخ اصلی والد چندین نخ (یک یا بیشتر) نخ فرزند تولید می کند (اصطلاحاً عمل fork انجام می شود) و سپس نخ والد تا اتمام نخ های فرزند منتظر می ماند. در نهایت پس از خاتمه نخ های فرزند، نخ اصلی به منظور بازیابی و ترکیب نتایج حاصل به نخ های فرزندی پیوندد (اصطلاحاً عمل join انجام می شود).

این مدل، یک مدل همگام به حساب می آید و می تواند برای ایجاد هر دو نوع نخ صریح (explicit) و ضمنی (implicit) به کار گرفته شود. شکل زیر موازی سازی Fork-Join را نشان می دهد



موازی سازی Fork-Join

۲۳ ذخیره سازی محلی نخ (Thread-Local Storage)

همانطور که پیشتر اشاره کردیم، نخ‌های متعلق به یک فرآیند، بخش داده (Data) یک فرآیند را به اشتراک دارند و این ویژگی را به عنوان یکی از مزایای برنامه‌نویسی چندنخی در نظر گرفتیم. با این حال در برخی شرایط، هر نخ ممکن است نیاز به داشتن یک کپی خاص خود از داده‌های مشخصی داشته باشد. به چنین نگهداری داده‌ای ذخیره سازی محلی نخ (یا TLS) اطلاق می‌شود.

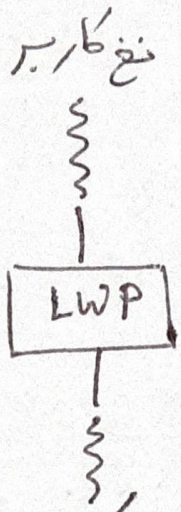
مثال برای استفاده از TLS: یک سیستم ترانکشن - پردازشی را در نظر بگیرید که در آن سرویس‌دهنده برای هر ترانکشن از یک نخ مجزا استفاده می‌شود. از طرفی فرض کنید که هر ترانکشن در این سیستم یک شناسه یکتا دارد. در چنین سیستمی برای اختصاص نخ به شناسه یکتای ترانکشن مربوطه می‌توانیم از TLS بهره بگیریم چرا که این شناسه خاص یک ترانکشن و یک نخ مشخص است.

امکان انجام در رابطه با TLS: در نگاه اول ممکن است TLS را با متغیرهای محلی اشتباه بگیریم. با این حال باید توجه کرد که متغیرهای محلی تنها در طول یک فراخوانی تابع قابل مشاهده هستند در حالیکه داده‌های TLS در طول کل فراخوانی‌ها قابل مشاهده هستند.

از برخی جهات TLS شبیه داده ایستا (static) می‌باشد با این تفاوت که داده TLS برای هر نخ یکتاست.

۲۴ فرآیندهای سبک وزن (Lightweight Processes)

در مدل‌های چند به چند و دوسطحی برای ارتباط بین نخ‌های سطح کاربر و سطح کرنل، تعداد زیادی از پیاده‌سازی‌ها، داده ساختار میانی را به نام فرآیند سبک وزن (Lightweight Process) در نظر می‌گیرند. شکل زیر ارتباط بین فرآیند سبک‌وزن و نخ‌های سطح کاربر و کرنل را نشان می‌دهد.



شکل ۶ فرآیند سبک وزن

فرآیند سبک وزن از دید نخ کاربر به عنوان یک پردازنده مجازی در نظر گرفته می‌شود که توسعه دهنده به هنگام توسعه برنامه کاربردی می‌تواند آنرا برای اجرای نخ‌های سطح کاربر زمانبندی نماید. از سوی دیگر، هر LWP متصل به یک نخ کرنل است و در واقع این نخ‌های کرنل هستند که توسط سیستم عامل برای اجرا روی پردازنده فیزیکی زمانبندی می‌شوند.

در چنین سناریویی، اگر نخ سطح کرنل بنا به هر دلیلی (مثل انتظار برای تکمیل I/O) بلاک شود، LWP مربوطه نیز بلاک خواهد شد و در پایان زنجیره نیز نخ سطح کاربر متصل به LWP مذکور نیز بلاک خواهد شد.