- خاتمه فرآیند توسط فرآیند پدر: یک فرآیند ممکن است با استفاده از فراخوان های سیستمی توسط فرآیند دیگری خاتمه یابد. به طور معمول چنین فراخوان هایی توسط فرآیند پدر (Parent) انجام می شود.

- خاتمه توسط کاربر یا برنامه کاربری فرابار: به چنین خاتمه ای اصطلاحاً Kill کردن یک فرآیند اطلاق می شود.

۴۰ ضرورت آگاهی شناسه فرآیند فرزند توسط فرآیند پدر: یک فرآیند اگر بخواهد فرآیند فرزند خود را خاتمه دهد باید شناسه آن را بداند. بنابراین وقتی یک فرآیند جدید به عنوان مثال با فراخوان سیستمی نظیر fork ایجاد می شود، شناسه آن فرآیند برای فرآیند پدر پیش ارسال می شود.

مطلب تکمیلی: فراخوانی fork توسط یک فرآیند، یک شناسه فرآیند (PID) است که خروجی حاصل از این به بعد فرآیند با شناسه PID می تواند مقادیر زیر را به خود بگیرد:

۱ PID>0 : فرآیند جدید به درستی ایجاد شده و معتبر است،

۲ PID<0 : شناسه نامعتبر بوده و فرآیند جدید با موفقیت ایجاد نشده است،

۳ PID=0 : آدرس فرآیند پدر و فرزند یکسان باشد، مقدار خروجی دستور fork برای فرآیند فرزند حاوی مقدار صفر است.
لازم به ذکر است که مقدار خروجی تابع fork برای فرآیند پدر، PID فرآیند جدید است.

۴۱ برخی از دلایل که فرآیند پدر سبب خاتمه فرآیند فرزند می شود عبارتند از:

۱ استفاده بیش از حد فرآیند فرزند از برخی منابع تخصیصی،

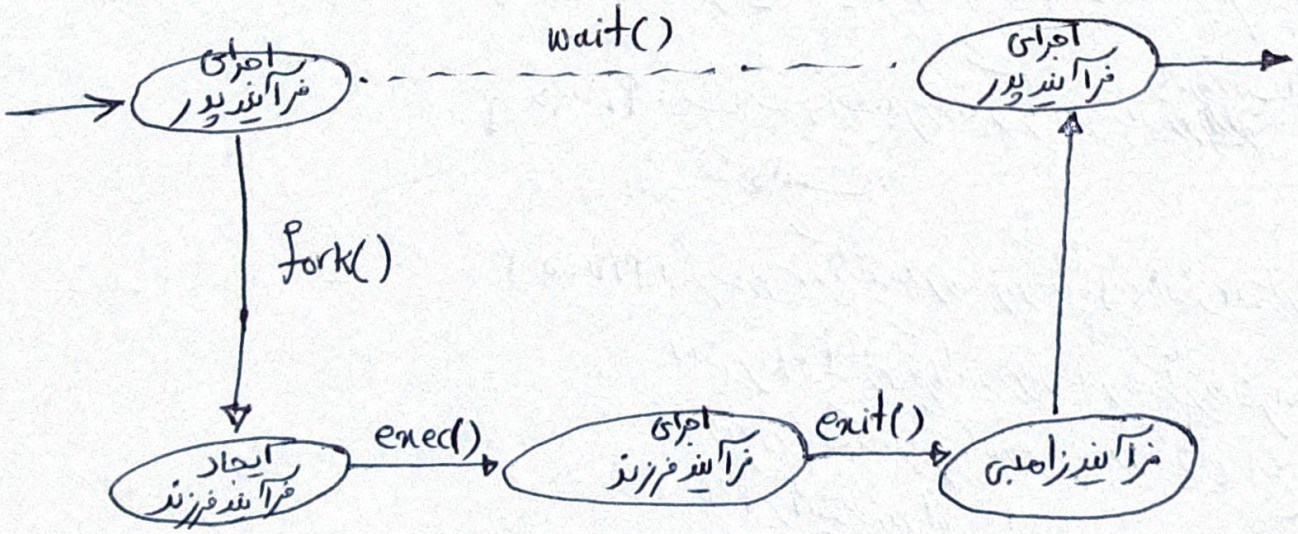
۲ عدم نیاز به وظیفه محول شده به فرآیند فرزند

۳ اتمام فرآیند پدر (به صورت نرمال / غیرنرمال) و عدم اجازه برخی سیستم عامل ها به ادامه حیات فرآیندهای فرزند

(۴۲) خاتمه آبشاری فرآیندها: به نحوه خاتمه‌ای که در آن اتمام فرآیند پدر موجب خاتمه تمامی فرآیندهای فرزندی می‌شود، خاتمه آبشاری (Cascading Termination) اطلاق می‌شود (cascading termination)

فرآیند زامبی (Zombie Process): هنگامی که یک فرآیند خاتمه پیدا می‌کند تا زمانیکه که خاتمه آن توسط فرآیند پدر شناسایی می‌شود، به فرآیند مذکور، فرآیند زامبی اطلاق می‌شود.

در واقع با خاتمه فرآیند، منابع تخصیص یافته به آن آزاد می‌شود ولی همچنان چنین فرآیندی در جدول فرآیندها باقی می‌ماند تا فرآیند پدر از وضعیت آن مطلع شود. شکل زیر مفاهیم بالا را به صورت مصور نشان می‌دهد.



۴۳) سؤال: فرض کنید فرآیند پدر بنا به هر دلیلی خاتمه یابد (شکل قبل را در نظر بگیرید). در چنین وضعیتی فرآیندهای فرزند مربوط به این فرآیند به طور خاص دیگر فاقد فرآیند پدر در سیستم هستند و اصطلاحاً به آنها فرآیندهای یتیم (Orphan Process) اطلاق می‌شود. حال سؤال این است که در چنین وضعیتی آزاد کردن مداخل جدول فرآیند و آزادسازی فرآیندهای یتیم بر عهده چه کسی است؟

پاسخ: در سیستم عامل‌ها به طور معمول چنین حالتی را با در نظر گرفتن فرآیند با PID=1 (فرآیند systemd) به عنوان پدر فرآیندهای یتیم کنترل و مدیریت می‌کنند. بدین نحو که فرآیند مذکور با فراخوانی تابع wait به طور دوره‌ای سبب آزادسازی فرآیندهای یتیم و آزاد کردن مداخل جدول فرآیندها می‌شود.

۴۴) مطلب تکمیلی- دلایل خاتمه فرآیند و استراتژی‌های مربوط به آن به صورت گسترده‌ای در سیستم عامل‌های مختلف قابل رده‌بندی هستند. به عنوان مثال در سیستم عامل‌های اندروید برای فرآیندها سلسله مراتبی از اولویت‌ها لحاظ می‌شود که بر اساس آن مشخص می‌شود به منظور محدودیت حافظه، فرآیندهایی برای خاتمه و آزادسازی منابع انتخاب شوند که اولویت کمتری دارند. سلسله مراتب یاد شده به ترتیب از اولویت بالا به اولویت پایین به قرار زیر است:

۱. فرآیندهای پیش زمینه (Foreground Process): فرآیندهای جاری قابل مشاهده روی صفحه که کاربر در حال حاضر با آنها تعامل دارد.

۲. فرآیندهای قابل مشاهده (Visible Process): فرآیندهایی که به طور مستقیم روی پیش زمینه قابل مشاهده نیستند اما در حال انجام فعالیتی

هستند که فرآیندهای پیش زمینه به آنها ارجاع می کنند (مثلاً فرآیندی که در حال انجام فعالیت و وضعیت آن روی صفحه نمایش داده می شود).

۳ فرآیندهای سرویس (Service Process): شبیه فرآیندهای پس زمینه هستند با این تفاوت که در حال انجام فعالیتی هستند که قابل مشاهده توسط کاربر است (مثلاً فرآیند پخش موسیقی).

۴ فرآیندهای پس زمینه (Background Process): فرآیندهایی هستند که در حال انجام فعالیتی می باشند که از دید کاربر پنهان است.

۵ فرآیند تهی (Empty Process): فرآیندی است که هیچگونه مؤلفه فعال (Active Component) تخصیص داده نشده به برنامه کاربردی ندارد.

نکته: فرآیندها در چنین سیستم هایی ممکن است بطور همزمان چند دیگر با اولویت باشند به عنوان مثال هم visible باشند هم service باشند، در این صورت اولویت در نظر گرفته برای آنها بالاترین سطح (یعنی visible) خواهد بود.

(۲۸) ارتباطات داخلی فرآیندها (IPC = InterProcess communication)

فرآیندهای موجود در سیستم را می توان از کاملا همرونی به دو گروه زیر تقسیم بندی کرد:

① فرآیندهای مستقل (Independent): به فرآیندهایی اطلاق می شود که با هیچ فرآیند در حال اجرای دیگری داده ای را به اشتراک نمی گذارند.

(۲۸) فرآیندهای همکار (cooperate): هر فرآیندی که اشتراک گذاری داده با دیگر فرآیندها داشته باشد، یک فرآیند همکار شناخته می‌شود.

در یک تعریف کلی‌تر، فرآیندهای همکار به فرآیندهایی اطلاق می‌شود که بتوانند بر اجرای دیگر فرآیندها در حال اجرا تاثیر بگذارند یا روی آن‌ها تاثیر بگذارند.

(۲۹) برخی دلایل برای تامین یک محیط همکاری بین فرآیندی عبارتند از:

① اشتراک گذاری اطلاعات (Information sharing): وجود انگیزه در چندین برنامه کاربردی برای دسترسی همزمان به یک قطعه اطلاعاتی مشترک،

② تسریع محاسبات (Computation speedup): اگر یک تسک را بخواهیم سریعتر اجرا کنیم می‌بایست آن را به زیرتسک‌هایی شکسته و هرکدام را به طور موازی با یکدیگر اجرا کنیم. برای یکپارچگی و عملکرد صحیح سیستم در چنین شرایطی نیاز مند محیط کاری مشترک برای کنترل و مدیریت زیرتسک‌ها و در نهایت تسک اصلی هستیم.

③ ماژولار یا پیمانه‌ای بودن (Modularity): ممکن است در سیستم تمایل داشته باشیم که توابع سیستم را در چندین دسته از فرآیندها یا نخ‌های مجزا تقسیم‌بندی کنیم در چنین حالتی برای داشتن عملکرد صحیح و یکپارچه داشتن محیط همکاری مشترک ضروری است.

(۳۰) فرآیندهای cooperate که منظور تبادل داده (ارسال/دریافت داده به/از فرآیندهای دیگر) نیاز به ارتباط درون فرآیندی (IPC) دارند. دو مدل پایه برای چنین