

سوالات تشریحی

(1)

انتقال داده در `named pipe` و `unnamed pipe` به یک شکل است، در زمان نوشتن داده از حافظه اختصاصی نویسنده به بافر سرکولار کرنل کپی میشود و در زمان خواندن نیز همین اتفاق بلعکس میفتد. و تفاوت این دو پایپ در نقطه شروع یا همان راه اندازی اشان است. `Unnamed pipe` کرنل بافر را میسازد و آدرس `fd` آن را تنها به پراسس های پدر و فرزند میدهد که شکل وراثتی دارد.

در حالی که در `named pipe` آدرس کرنل بافر تولید شده به صورت یک فایل در دایرکتوری قرار میگیرد، و در نتیجه هر کس میتواند آن را باز کند اما هر دو همچنان به یک بافر حلقوی در کرنل اشاره میکنند. در نتیجه مقصد هر دو یکیست اما در نحوه اتصال متفاوت اند.

(2)

در مرحله ذخیره سازی از آنجایی که هر دو آنها مقصدی یکسان دارند سرعت تقریباً یکسانی دارند، اما در فاز های ایجاد و همچنین اتصال متفاوت اند.

`Named pipe` از آن جایی که ماهیت یک فایل سیستم را دارد، برای باز شدن نیاز دارد تا تمامی اورهد های مربوطه مانند چک کردن نام فایل، دسترسی ها و ... را طی کند در حالی که `unnamed pipe` مستقیماً توسط کرنل ایجاد میشود و هیچ یک از این سربار هارا ندارد.

`Named pipe` برای ایجاد نیاز به یک سیستم کال `mkfifo` و دوبار `open` برای نویسنده و خواننده است، در حالی که `unnamed pipe` تنها یک فراخوانی `pipe` دارد، و پس از آن دو `fd` در اختیار ما قرار میدهد.

در مورد زمان بهره وری CPU نیز میتوان گفت از آنجایی که باز هم `named pipe` ماهیت یک فایل سیستم را دارد تمامی حالاتی که در مدیریت فایل ها و همچنین کانتکست سوئیچ میتواند رخ دهد برای آن نیز صدق میکند، پس میتوان گفت باز هم ممکن است سیکل بیشتری مصرف کند.

در نتیجه میتوان گفت `unnamed pipe` سریع تر است، چرا که برای استفاده میان دو پردازنده پدر و فرزند است که کاملاً مرتبط اند و ماهیت موقتی نیز دارد، درحالی که `named pipe` برای ارتباط میان پردازنده های نه چندان مرتبط نیز استفاده میشود.

(3)

همانطور که اشاره شد، `named pipe` میتواند میان پردازنده های مختلف ارتباط برقرار کند پس انعطاف پذیری بالاتری دارد، با توجه به هزینه بیشترش.

Unnamed pipe پس از خاتمه یافت پردازش هایش از بین میرود، چرا که همانطور که گفته شد مزیت آن برای کارهای موقت است. درحالی که named pipe تا زمانی که unlink نشود باقی می ماند چرا که همچنان یک اشاره گر به آن در حافظه وجود دارد.

در نتیجه میتوان گفت اگر روابط وراثتی در پردازش وجود داشته باشد، بهترین انتخاب unnamed pipe است مانند حالت | در پروژه ما، اما اگر قرار باشد بخواهیم ارتباطی با پردازش دیگری داشته باشیم، چاره ای بجز استفاده از named pipe نداریم.

مقایسه عملی

Command: unnamed.sh	Command: named.sh
Average User time (seconds): 1.949	Average User time (seconds): 2.735
Average System time (seconds): 1.135	Average System time (seconds): 1.422
Average CPU %: 130.80	Average CPU %: 87.70
Average Elapsed time (seconds): 2.357	Average Elapsed time (seconds): 4.707
Average File system inputs: 0.000	Average File system inputs: 0.000
Average File system outputs: 0.000	Average File system outputs: 0.000

Unnamed pipe همانطور که دیده شد بسیار سریع تر است. چرا که درون اسکریپت named.sh سربار فایل سیستم بسیار بیشتری وجود دارد (1000 بار اجرای دستورات rm , mkfifo) در حالی که در پایپ بی نام، فایل سیستم اصلاً درگیر نمیشود و صرفاً دو بافر در RAM تشخیص میدهد.

System Time: همانطور که در بخش های قبلی گزارش گفته شد، پایپ بی نام تنها از یک فراخوانی pipe() برای ساخت استفاده میکند و به ما دو FD تحویل میدهد، درحالی که در پایپ نام دار سیستم عامل باید فراخوانی های mkfifo و open و unlink را برای مدیریت کند که باعث افزایش زمان در آن میشود.

CPU Usage: دلیل آنکه برای پایپ بی نام این مقدار 130% میباشد این است که بهره وری بهتری در استفاده از چند هسته صورت گرفته، چرا که سربار یا block بخصوصی اتفاق نمی افتد تا جلوی هسته ها گرفته شود.

این درحالیست که در پایپ نام دار، cpu در بخش از زمان خود بصورت Idle و یا Wait بوده چرا که مدیریت فایل در اینجا دخیل است مانند blocking در فراخوانی open پس هر بار در حلقه سیستم باید صبر کند تا یک فایل ساخته شود و خواننده به آن وصل شود. که تمامی این ها باعث عدم استفاده از توان CPU میشود.