Pengertian Interprocess Communication

Interprocess Communication (IPC) adalah satu set antarmuka pemrograman yang memungkinkan programmer untuk mengkoordinasikan kegiatan antara proses program yang berbeda yang dapat dijalankan secara bersamaan dalam sebuah sistem operasi. Hal ini memungkinkan program untuk menangani banyak permintaan pengguna pada saat yang sama. Karena bahkan permintaan pengguna tunggal dapat mengakibatkan beberapa proses yang berjalan di sistem operasi atas nama pengguna, proses harus berkomunikasi satu sama lain. IPC interface membuat ini mungkin. Setiap metode IPC memiliki kelebihan dan keterbatasan sendiri sehingga tidak biasa untuk satu program untuk menggunakan semua metode IPC.

Pengertian lain dari IPC adalah proses komunikasi yang digunakan antara program dan proses yang berjalan di sistem operasi multitasking atau antar komputer jaringan.

Ada dua jenis IPC (Interprocess Communication)

*LPC* (panggilan prosedur lokal) LPCs digunakan dalam sistem operasi multitasking untuk memungkinkan tugas secara bersamaan berjalan untuk berbicara satu sama lain. Mereka bisa berbagi ruang memori, sinkronisasi tugas, dan mengirim pesan ke satu sama lain.

*RPC* (remote procedure call) RPC mirip dengan LPC tapi pekerjaan melalui jaringan. RPC menyediakan mekanisme yang klien gunakan untuk berkomunikasi permintaan untuk layanan untuk sistem network attached lain seperti server. Jika Anda berpikir dari aplikasi client / server sebagai program yang telah dibagi antara front-end dan back-end sistem, RPC dapat dilihat sebagai komponen yang mengintegrasikan mereka melalui jaringan. RPC kadang-kadang disebut mekanisme kopling.

|  |
| --- |
| [Pengertian Interprocess Communication](http://1.bp.blogspot.com/-qwiHrf2Ox2I/VqxwWu61VyI/AAAAAAAABCA/77AEDGE01tg/s1600/Interprocess+Communication.png) |
| Interprocess Communication |

Keuntungan IPC

Salah satu keuntungan menggunakan IPC adalah bahwa program dapat mengambil keuntungan dari proses ditangani oleh program atau komputer lain. Model client / server mengambil keuntungan dari RPC, seperti yang didistribusikan arsitektur objek seperti*Microsoft DCOM* (Distributed Component Object Model) dan *CORBA* (Common Object Request Broker Architecture).  
Mekanisme komunikasi interprocess normal dalam UNIX adalah pipa, dan soket adalah mekanisme komunikasi interprocess yang bekerja di jaringan. Ini menjadi bagian dari UNIX ketika TCP / IP stack protokol diintegrasikan ke Berkeley UNIX di awal 1980-an. Ini adalah proyek yang didanai oleh *DARPA*.

Fitur IPC

Fitur IPC dijelaskan di sini digunakan jika program perlu berkomunikasi dengan satu sama lain atau berbagi daerah memori: *Shared Proses memori* dapat bertukar nilai dalam memori bersama. Memori menjadi semacam papan buletin di mana proses dapat mengirim informasi status dan data yang perlu dibagi. *Antrian Sebuah antrian* IPC adalah daftar terstruktur dan memerintahkan segmen memori di mana proses menyimpan atau mengambil data.

*Semaphore* Sebuah semaphore menyediakan mekanisme sinkronisasi untuk proses yang mengakses sumber daya yang sama. Tidak ada data dilewatkan dengan semaphore-itu hanya koordinat akses ke sumber daya bersama. Pipa pipa A menyediakan cara untuk proses untuk berkomunikasi dengan satu sama lain dengan bertukar pesan. Pipa bernama menyediakan cara untuk proses yang berjalan pada sistem komputer yang berbeda untuk berkomunikasi melalui jaringan. Slot mail adalah sistem pesan-toko dan-maju yang tidak memerlukan stasiun untuk melakukan sinkronisasi dengan satu sama lain.

Berikut ini adalah karakteristik dari *Interprocess Communincation*(IPC) :

1. komunikasi *Synchronous* dan *asynchronous*

pada komunikasi *Synchronous*, proses sinkronisasi pada pengiriman dan penerimaan terjadi pada setiap pesan dan sistem ini akan bekerja secara bergantian, jika sistem sedang mengirim pesan, maka sistem hanya akan boleh merespon, sampai pesan tersebut sampai. Pada komunikasi *asynchronous*, komunikasi ini dapat segera memproses sebuah pesan, setelah pesan tersebut berada di local buffer, dan pemrosesan pesan dengan pengiriman pesan bekerja secara peralalel

1. Message destinations

tempat tujuan dari sebuah pesan yang terdapat pada computer adalah local port, yang didefinisikan sebagai variable angka dengan tipe integer. Sebuah port pasti mempunyai satu penerima, akan tetapi bisa memiliki banyak pengirim.

1. Reliability

Kehandalan dari sebuah sistem dapat dilihat dari validity dan integrity dari sistem itu.

Sistem jika dilihat dari validty, dapat dikatakan handal jika, pesan yang disampaikan dijamin sampai tanpa ada pesan yang hilang atau jatuh, begitu juga sebaliknya.

1. Ordering

menginginkan pesan yang terkirim dari pengirim dapat diterima sesuai dengan urutan *grouping / ordering*berdasarkan pesan awal yang terikirim.

Kategori dari IPC

1. Shared memory

Sistem Berbagi Memori atau yang disebut juga sebagai Shared Memory System merupakan salah satu cara komunikasi antar proses dengan cara mengalokasikan suatu alamat memori untuk dipakai berkomunikasi antar proses. Alamat dan besar alokasi memori yang digunakan biasanya ditentukan oleh pembuat program. Pada metode ini, sistem akan mengatur proses mana yang akan memakai memori pada waktu tertentu sehingga pekerjaan dapat dilakukan secara efektif.

1. Pipe

Pipe merupakan komunikasi sequensial antar proses yang saling terelasi, namun pipe memiliki kelemahan yaitu hanya bisa digunakan untuk komunikasi antar proses yang saling berhubungan, dan komunikasinya yang dilakukan adalah secara sequensial. Urutan informasi yang ada dalam sebuah pipe ada yang mirip dengan antrian queue. Jika komunikasi yang diinginkan adalah komunikasi dua arah maka kita harus membuat dua pipe, karena sebuah pipe hanya bisa digunakan untuk komunikasi satu arah saja.

1. Messages passing

Sistem berkirim pesan adalah proses komunikasi antar bagian sistem untuk membagi variabel yang dibutuhkan. Proses ini menyediakan dua operasi yaitu mengirim pesan dan menerima pesan. Ketika dua bagian sistem ingin berkomunikasi satu sama lain, yang harus dilakukan pertama kali adalah membuat sebuah link komunikasi antara keduanya. Setelah itu, kedua bagian itu dapat saling bertukar pesan melalui link komunikasi tersebut. Sistem berkirim pesan sangat penting dalam sistem operasi.

1. Message Queue,

adalah sebuah data stream anonim yang mirip dengan metode Pipe, tetapi Message Queue menyimpan dan mengambil informasi dalam bentuk packets. Metode ini dapat dijalankan di kebanyakan OS.

1. Named Pipe,

adalah implementasi Pipe melalui sebuah file pada sistem file, bukan melalui standar input dan output. Metode ini dapat dijalankan di semua sistem yang mendukung POSIX (Portable Operating System Interface), Windows.

1. Semaphore,

adalah sebuah struktur sederhana yang mensinkronisasikan thread atau proses yang bekerja pada sumber daya yang sama. Metode ini dapat dijalankan di semua sistem yang mendukung POSIX (Portable Operating System Interface), Windows.

1. Memory-mapped File,

adalah sebuah file yang dipetakan ke RAM dan dapat dimodifikasi dengan mengubah alamat memori secara langsung. Metode ini dapat dijalankan di semua sistem yang mendukung POSIX (Portable Operating System Interface), Windows.

Terdapat dua macam cara berkomunikasi antar proses, yaitu:

1.       Komunikasi langsung

Setiap proses yang ingin berkomunikasi harus memiliki nama yang bersifat eksplisit  baik penerimaan atau pengirim dari komunikasi tersebut. Dalam konteks ini, pengiriman dan penerimaan pesan secara primitive dapat dijabarkan sebagai :

* Send (P, message) – mengirim sebuah pesan ke proses P.
* Receive (Q, message) – menerima sebuah pesan dari proses Q.

Sebuah jaringan komunikasi pada bahasan ini memiliki beberapa sifat, yaitu:

* Sebuah jaringan yang didirikan secara otomatis diantara setiap pasang dari proses yang ingin dikomunikasikan. Proses tersebut harus mengetahui identitas dari semua yang ingin dikomunikasikan.
* Sebuah jaringan adalah terdiri dari penggabungan dua proses.
* Diantara setiap pesan dari proses terdapat tepat sebuah jaringan.  
  Pembahasan ini memperlihatkan sebuah cara simetris dalam pemberian alamat. Oleh karena itu, baik keduanya yaitu pengirim dan penerima proses harus memberi nama bagi yang lain untuk berkomunikasi, hanya pengirim yang memberikan nama bagi penerima sedangkan penerima tidak menyediakan nama bagi pengirim. Dalam konteks ini, pengirim dan penerima secara sederhana dapat dijabarkan sebagai berikut:
* Send (P, message) – mengirim sebuah pesan kepada proses P.
* Receive (id, message) – menerima sebuah pesan dari semua proses. Variabel id diatur sebagai nama dari proses dengan komunikasi.

2.       Komunikasi tidak langsung

Dengan komunikasi tidak langsung, pesan akan dikirimkan pada dan diterima dari/ melalui mailbox (kotak surat) atau terminal-terminal, sebuah mailbox dapat dilihat secara abstrak sebagai sebuah objek didalam setiap pesan yang dapat ditempatkan dari proses dan dari setiap pesan yang bias dipindahkan. Setiap kotak surat memiliki sebuah identifikasi (identitas) yang unik, sebuah proses dapat berkomunikasi dengan beberapa proses lain melalui sebuah nomor dari mailbox yang berbeda.

Dua proses dapat saling berkomunikasi apabila kedua proses tersebut sharing mailbox. Pengirim dan penerima dapat dijabarkan sebagai :

* Send (A, message) – mengirim pesan ke mailbox A.
* Receive (A, message) – menerima pesan dari mailbox A.

Dalam masalah ini, link komunikasi mempunyai sifat sebagai berikut :

* Sebuah link dibangun diantara sepasang proses dimana kedua proses tersebut membagi mailbox.
* Sebuah link mungkin dapat berasosiasi dengan lebih dari dua proses.
* Diantara setiap pasang proses komunikasi, mungkin terdapat link yang berbeda-beda, dimana setiap link berhubungan pada satu mailbox.  
  Misalkan terdapat proses P1, P2 dan P3 yang semuanya share mailbox. Proses P1 mengirim pesan ke A, ketika P2 dan P3 masing-masing mengeksekusi sebuah kiriman dari A. Proses mana yang akan menerima pesan yang dikirim P1? Jawabannya tergantung dari jalur yang kita pilih :
* Mengizinkan sebuah link berasosiasi dengan paling banyak 2 proses.
* Mengizinkan paling banyak satu proses pada suatu waktu untuk mengeksekusi hasil kiriman (receive operation).
* Mengizinkan sistem untuk memilih secara mutlak proses mana yang akan menerima pesan (apakah itu P2 atau P3 tetapi tidak keduanya, tidak akan menerima pesan). Sistem mungkin mengidentifikasi penerima kepada pengirim.  
  Mailbox mungkin dapat dimiliki oleh sebuah proses atau sistem operasi. Jika mailbox dimiliki oleh proses, maka kita mendefinisikan antara pemilik (yang hanya dapat menerima pesan melalui mailbox) dan pengguna dari mailbox (yang hanya dapat mengirim pesan ke mailbox).

Selama setiap mailbox mempunyai kepemilikan yang unik, maka tidak akan ada kebingungan tentang siapa yang harus menerima pesan dari mailbox. Ketika proses yang memiliki mailbox tersebut diterminasi, mailbox akan hilang. Semua proses yang mengirim pesan ke mailbox ini diberi pesan bahwa mailbox tersebut tidak lagi ada. Dengan kata lain, mempunyai mailbox sendiri yang independent, dan tidak melibatkan proses yang lain. Maka sistem operasi harus memiliki mekanisme yang mengizinkan proses untuk melakukan hal-hal dibawah ini:

- Membuat mailbox baru.

-Mengirim dan menerima pesan melalui mailbox.

-Menghapus mailbox.

Proses yang membuat mailbox pertama kali secara default akan memiliki mailbox tersebut. Untuk pertama kali, pemilik adalah satu-satunya proses yang dapat menerima pesan melalui mailbox ini.

1. **Implementasi Masalah Interprocess Communication pada Sistem Operasi LINUX**

Proses berkomunikasi antara satu dengan proses – proses lainnya dan dengan kernel untuk mengkoordinasikan kegiatan setiap proses. Linux mendukung sejumlah mekanisme dari Inter-Process Communication (IPC). Contoh mekanisme Inter-Process Communication (IPC) adalah mekanisma *Signal* dan *pipe*. Linux juga mendukung mekanisme Sistem V IPC.

**Masalah terjadinya concurrent access pada kernel linux**

ada suatu saat dalam sebuah kernel, tidak terkecuali kernel LINUX, dapat terjadi akses bersamaan (*concurrent access*). Dalam hal ini diperlukan proteksi dalam kernel yang bersangkutan. Proteksi dapat dilakukan dengan sinkronisasi.

Sebuah proses memiliki bagian dimana bagian ini akan melakukan akses dan manipulasi data. Bagian ini disebut dengan *critical section*. Ketika sebuah proses sedang dijalankan dalam critical section nya, tidak ada proses lain yang boleh dijalankan dalam critical section nya.

Ada dua jenis concurrency yaitu pseudo-concurrency dan true-concurrency. Ada beberapa penyebab konkurensi kernel, diantaranya interrupt, softirqs dan tasklets, kernel preemption,sleeping dan synchronization with user-space, dan symmetrical multiprocessing.

Salah satu metode dalam kernel LINUX untuk sinkronisasi adalah atomic operations. Integer atomik adalah salah satu jenis dari atomic operations. Integer Atomik menyediakan instruksi yang dijalankan secara atomik (tanpa interrupt).

Locking yang paling umum digunakan dalam LINUX adalah spin lock. Spin lock adalah lock yang hanya dapat dilakukan oleh satu thread. Ketika sebuah thread yang akan dijalankan meminta spin lock yang sedang digunakan, maka thread ini akan loops menunggu sampai spin lock tersebut selesai digunakan oleh thread yang sedang berjalan.

Semafor dalam LINUX adalah sleeping locks. Ketika sebuah thread meminta semafor yang sedang digunakan, maka semafor akan meletakkan thread tersebut dalam wait queue dan menyebabkan thread tersebut masuk status sleep.

Symmetrical multiprocessing (SMP) mendukung adanya pengeksekusian secara paralel dua atau lebih thread oleh dua atau lebih processor. Kernel LINUX 2.0 adalah kernel LINUX pertama yang memperkenalkan konsep SMP.

**Signal**

*Signal* merupakan salah satu metode komunikasi antar-proses yang sebelumnya digunakan digunakan oleh sistem operasi UNIX. Mereka menggunakannya untuk *signal Asynchronous envet* antara satu atau lebih proses. Sebuah sinyal dapat dihasilkan oleh keyboard interrupt atau kondisi kesalahan seperti proses mencoba untuk mengakses lokasi tidak ada dalam memori virtual. *Signal* juga digunakan oleh Shell untuk sinyal perintah kontrol kerja kepada *child process* merekea.

Ada sebuah set *signal* yang mendefinisikan bahwa kernel dapat menghasilkan atau dapat dihasilkan oleh proses lain dalam sistem, jika dan hanya jika mereka memiliki *privileges* ( Hak ) yang sesuai. Berikut adalah daftar set sistem sinyal menggunakan perintah kill ( kill –l) pada saya kotak Intel Linux ini memberikan:

Linux mengimplementasikan *signal* menggunakan informasi yang disimpan dalam *task\_struct* untuk proses tersebut. Jumlah *signal* yang didukung terbatas pada ukuran *word prosesor*. Proses dengan ukuran *word* 32 bit dapat memiliki 32 *signal* sementara prosesor 64 bit seperti Alpha AXP dapat memiliki hingga 64 *signal*. *signal* yang tertunda saat ini disimpan di *signal field* dengan *mask* *of blocked signal* dalam keadaan terblokir. Dengan pengecualian dari SIGSTOP dan SIGKILL, semua *signal* dapat diblokir. Jika sinyal diblokir dihasilkan, tetap *pending* sampai tidak diblokir ( *unblocked*). Linux juga menyimpan informasi tentang bagaimana setiap proses menangani *signal* mungkin dan ini diadakan dalam berbagai *array* *sigaction* yang ditunjuk oleh task\_struct untuk setiap proses. Antara lain mengandung baik alamat rutin yang akan menangani *signal* atau *flag*, yang memberitahu Linux bahwa proses tersebut ingin mengabaikan *signal* ini atau membiarkan kernel menanganinya sinyal untuk itu. Proses memodifikasi penanganan default sinyal dengan membuat panggilan sistem dan call ini mengubah *sigaction* untuk *signal* yang sesuai *blocked mask*.

*signal* tidak disampaikan kepada proses segera setelah *signal* itu dihasilkan., Mereka harus menunggu sampai proses tersebut berjalan kembali. Setiap kali sebuah proses keluar dari sistem panggilan *signal* dan *blocked field* diperiksa , jika ada sinyal *signal*, sekarang dapat disampaikan kepada proses. mungkin metode yang sangat diandalkan tapi setiap proses dalam sistem membuat panggilan sistem, misalnya untuk menulis karakter ke terminal, sepanjang waktu. Proses dapat memilih untuk menunggu sinyal jika mereka ingin, mereka tergantung di *state Interruptible* sampai sinyal disajikan. *Linux* *signal processing code* signal terlihat pada struktur *sigaction* untuk setiap sinyal diblokir saat ini.

Jika sebuah penangan sinyal diset ke tindakan default maka kernel akan mengatasinya. Penangan default sinyal SIGSTOP akan mengubah keadaan proses saat untuk Berhenti dan kemudian menjalankan scheduler untuk memilih sebuah proses baru untuk menjalankan. Aksi default untuk sinyal SIGFPE akan core dump proses dan kemudian menyebabkan itu untuk keluar. Atau, proses dapat menentukan handler sinyalnya sendiri. Ini adalah rutinitas yang akan dipanggil setiap kali sinyal dihasilkan dan struktur sigaction menyimpan alamat rutinitas ini. Kernel harus memanggil penanganan sinyal rutin proses dan bagaimana hal ini terjadi adalah prosesor khusus, namun semua CPU harus menghadapi kenyataan bahwa proses saat berjalan dalam mode kernel dan hanya akan kembali ke proses yang disebut kernel atau sistem rutin user mode. Masalah diselesaikan dengan memanipulasi stack dan register proses. Proses ini program counter diatur ke alamat rutin penanganan sinyal dan parameter untuk rutin ditambahkan ke frame panggilan atau berlalu dalam register. Ketika proses kembali ke pengoperasian tampak seolah-olah routine penangan sinyalnya dipanggil secara normal.