MODUL SISTEM OPERASI

(CCI210)

MODUL SESI 2

MANAJEMEN PROSES

DISUSUN OLEH

ADI WIDIANTONO, SKOM, MKOM.

UNIVERSITAS ESA UNGGUL

2020

MANAJEMEN PROSES
DALAM SISTEM OPERASI

A. Kemampuan Akhir Yang Diharapkan

Setelah mempelajari modul ini, diharapkan mahasiswa mampu :

1. Memahami konsep dasar dari sebuah proses dalam Sistem Operasi.
2. Memahami manajemen proses dalam Sistem Operasi
3. Mamahami tantangan dalam keamanan proses dalam Sistem Operasi

B. Uraian dan Contoh

1. Proses dalam Sistem Operasi

1.1. Apakah Proses itu?

Proses dapat didefinisikan sebagai :

- Sebuah program yang sedang dieksekusi
- Bagian dari program yang berjalan di komputer
- Entitas yang dapat ditetapkan dan dijalankan pada prosesor
- Sebuah unit kegiatan yang ditandai dengan pelaksanaan urutan instruksi, keadaan saat ini, dan sekumpulan sumber daya sistem yang terkait

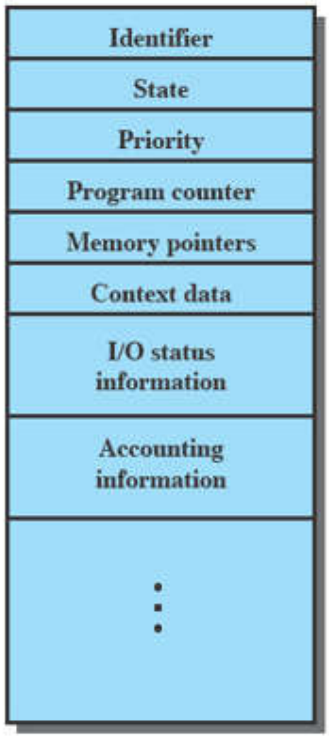
Sebuah proses dapat sebagai entitas yang terdiri dari sejumlah elemen, elemen esensial dari proses adalah **progam code** dan **sekumpulan data** yang berkaitan dengan code tersebut.

Selama dalam program dieksekusi, sebuah proses dapat secara unik dikarakteristikan dengan sejumlah elemen:

- Identifier/Pengenal: Pengenal unik yang terkait dengan proses ini, untuk membedakannya dari semua proses lainnya.
- Status: Jika proses sedang dijalankan, ini dalam status berjalan.
- Prioritas: Tingkat prioritas relatif terhadap proses lainnya.
- Program counter / Penghitung program: Alamat dari instruksi berikutnya dalam program yang akan dieksekusi.
- Memori Pointer/Penunjuk memori: Berisi penunjuk ke kode program dan data yang terkait dengan proses ini, ditambah blok memori apa pun yang dibagikan dengan proses lain.

- Data konteks: Ini adalah data yang ada dalam register di prosesor saat proses sedang dijalankan.
- Informasi status I / O: Termasuk permintaan I / O yang luar biasa, perangkat I / O (misalnya, disk drive) yang ditetapkan untuk proses ini, daftar file yang digunakan oleh proses tersebut, dan begitu seterusnya.
- Informasi akuntansi: Mungkin termasuk jumlah waktu dan jam prosesor waktu yang digunakan, batas waktu, nomor rekening, dan sebagainya.

Informasi karakteristik proses dalam daftar di atas disimpan dalam struktur data, yang disebut ***process control block.***



Gambar 1.1. Simplified Process Control Block

Blok kontrol proses adalah alat utama yang memungkinkan OS mendukung banyak proses dan menyediakannya multiprocessing. Ketika suatu proses terputus, nilai-nilai program saat ini, counter dan register prosesor (data konteks) disimpan di bidang yang sesuai dari blok kontrol proses yang sesuai, dan status proses diubah ke nilai lain, seperti diblokir atau siap berjalan (dijelaskan selanjutnya). OS bebas untuk menempatkan beberapa proses lain dalam status berjalan. Penghitung program dan data konteks untuk proses ini dimuat ke register prosesor dan mulai mengeksekusi.

Jadi, kita dapat mengatakan bahwa suatu proses terdiri dari kode program dan data terkait ditambah blok kontrol proses. Untuk komputer prosesor tunggal, pada waktu tertentu, hanya satu proses sedang dijalankan dan proses itu dalam status berjalan.

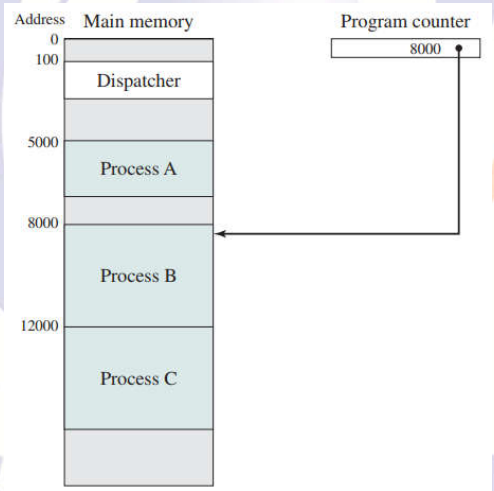
2. Tahapan Proses (Process States)

Dari pandangan sebuah processor, dalam menjalankan sebuah proses, terdapat dua hal penting yaitu ***trace*** dan ***dispatcher.***

Trace adalah daftar urutan instruksi yang dijalankan untuk proses. Dari sisi prosesor, dapat memperlihatkan dengan menunjukkan bagaimana jejak/trace dari berbagai proses disisipkan (interleaved).

Dispatcher adalah program kecil yang mengalihkan prosesor dari satu proses ke proses lainnya.

Contoh dari sebuah proses dengan trace dan dispatcher dapat dilihat dalam gambar berikut ini.



Gambar 2.1. Sepotong gambaran eksekusi proses

(a) Trace of process A	(b) Trace of process B	(c) Trace of process C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

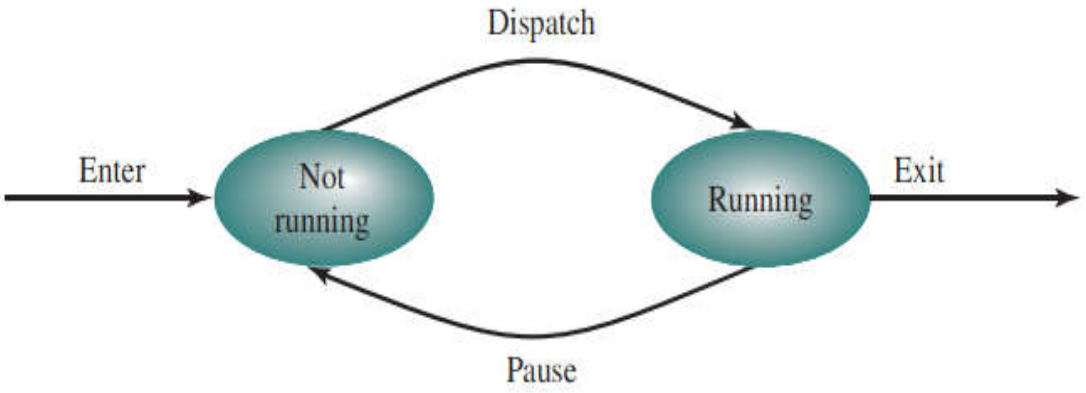
5000 = Starting address of program of process A
8000 = Starting address of program of process B
12000 = Starting address of program of process C

Gambar 2.2.Trace

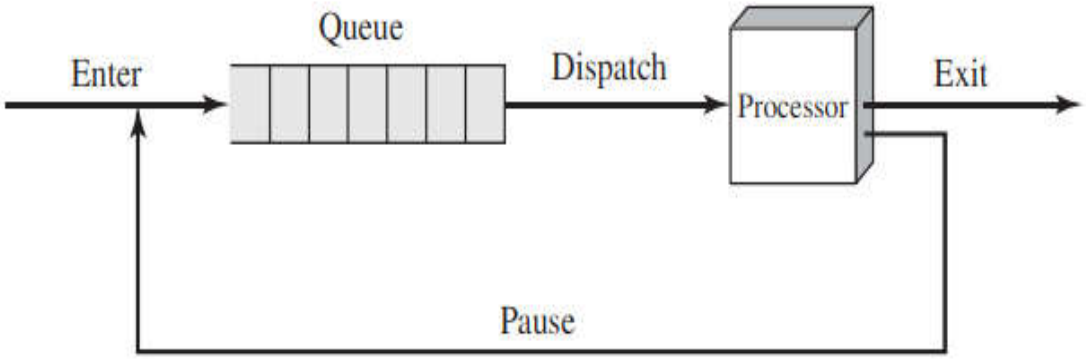
2.1. Two-State Model Process

Two states: Running or Not Running,

Ketika OS membuat proses baru, akan membuat blok kontrol proses untuk proses dan memasukkan proses itu ke dalam sistem dalam status *Not Running* (Tidak Berjalan). Prosesnya muncul, dan dikenal oleh OS, dan menunggu kesempatan untuk dieksekusi. Dari waktu ke waktu, proses yang sedang berjalan akan diinterupsi dan *dispatcher* OS akan memilih beberapa proses lain untuk dijalankan. Proses sebelumnya bergerak dari status Running (berjalan) ke status Not Running (Tidak Berjalan), dan salah satu proses lainnya dipindahkan ke status Running.



(a) State diagram



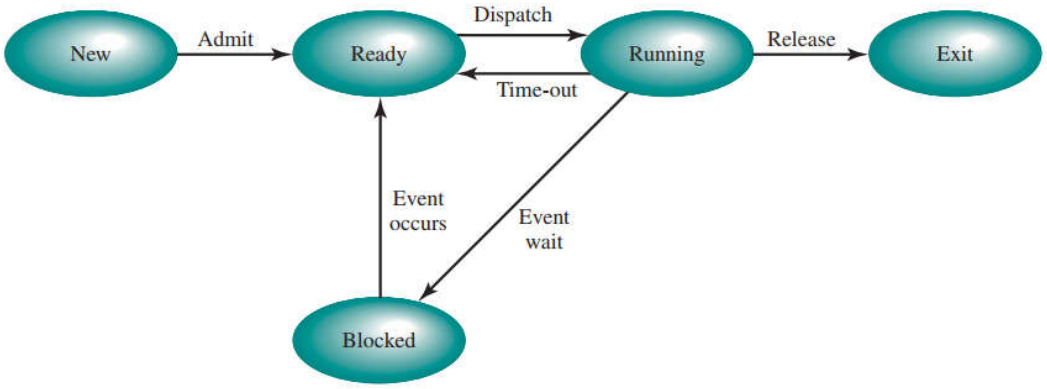
(b) Queing diagram

Gambar 2.3. Two-State Model

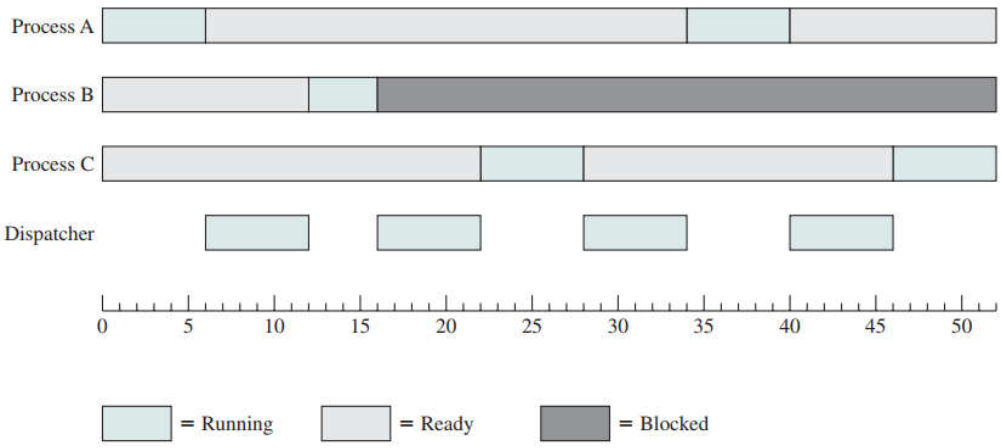
2.2. Five-State Model Process

Lima Status terdiri dari

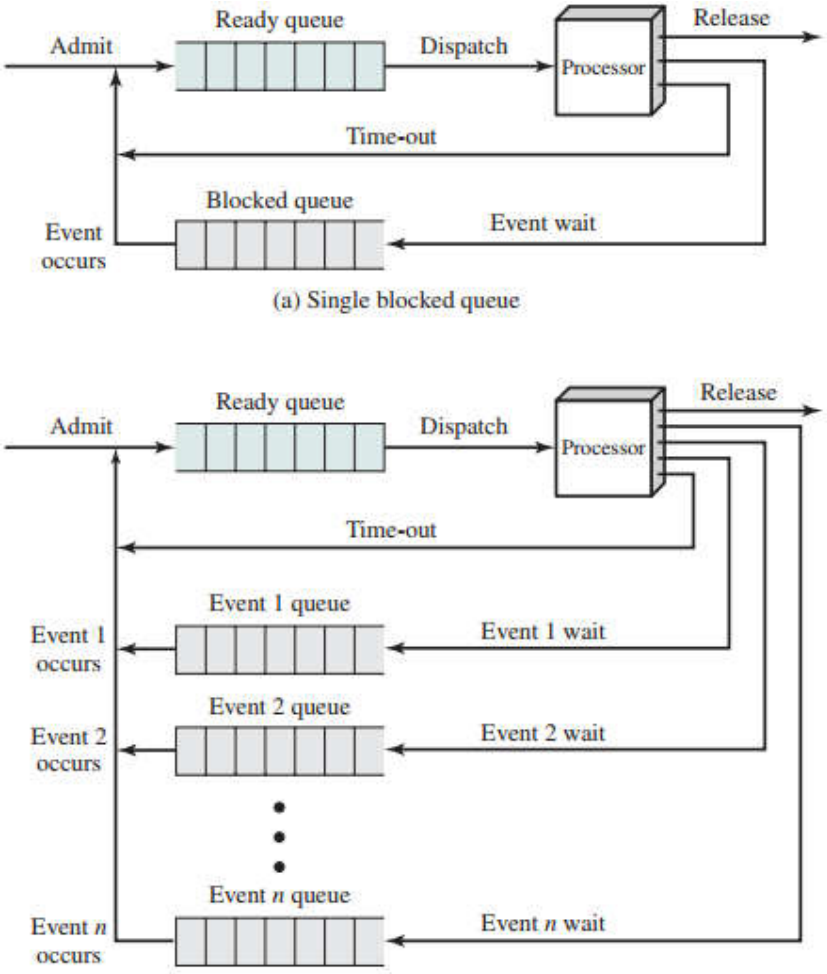
- Running: Proses yang saat ini sedang dijalankan. untuk prosesor tunggal, paling banyak satu proses pada suatu waktu bisa dalam keadaan ini.
- Ready: Proses yang disiapkan untuk dieksekusi saat diberi kesempatan.
- Blocked/Waiting: Proses yang tidak dapat dijalankan hingga beberapa peristiwa terjadi, seperti penyelesaian operasi I / O.
- New: Proses yang baru saja dibuat tetapi belum diterima di kumpulan proses yang dapat dieksekusi oleh OS. Biasanya, belum dimuat ke memori utama, meskipun blok kontrol prosesnya telah dibuat.
- Exit: Sebuah proses yang telah dilepaskan dari kumpulan proses yang dapat dieksekusi oleh OS, entah karena dihentikan atau karena dibatalkan karena suatu alasan.



Gambar 2.4. Five-State Model Process



Gambar 2.5. Proses Status dan Trace



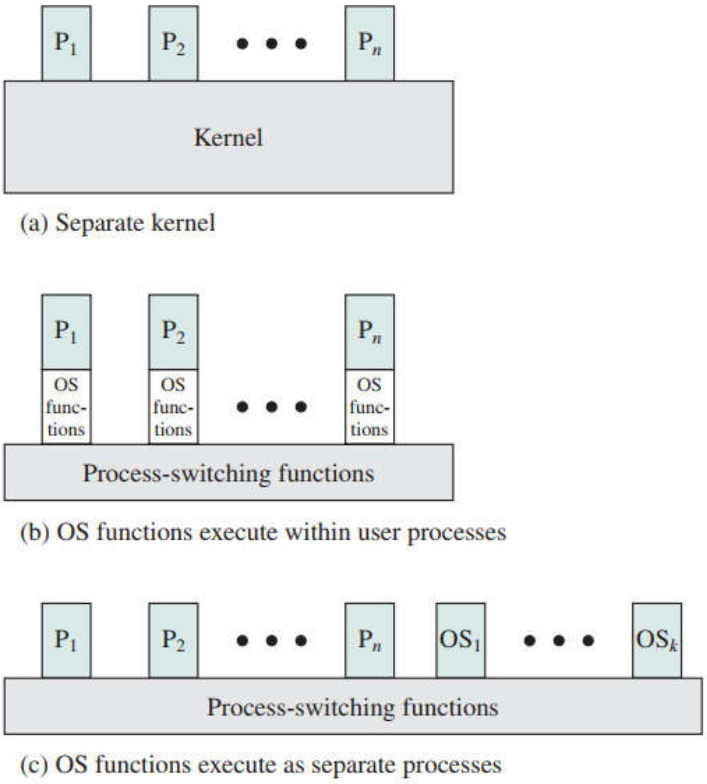
Gambar 2.5. Model Antrian

3. Eksekusi Proses

3.1. Nonprocess Kernel

Salah satu pendekatan tradisional, pada banyak sistem operasi lama, adalah dengan mengeksekusi kernel OS di luar proses apa pun (Gambar 3.1a). Dengan pendekatan ini, ketika proses yang sedang berjalan terputus atau mengeluarkan panggilan supervisor, file konteks mode dari proses ini disimpan dan kontrol diteruskan ke kernel. OS memiliki wilayah memorinya sendiri untuk digunakan dan tumpukan sistemnya sendiri untuk prosedur pengendalian panggilan dan kembali. OS dapat melakukan fungsi apa pun yang diinginkan dan memulihkan konteks proses yang terputus, yang menyebabkan eksekusi dilanjutkan dalam gangguan proses pengguna. Alternatifnya, OS dapat menyelesaikan fungsi penyimpanan proses dan melanjutkan untuk menjadwalkan dan mengirimkan proses lain. Hal utama di sini

adalah bahwa konsep proses dipertimbangkan hanya berlaku untuk program pengguna. Kode sistem operasi dijalankan sebagai entitas terpisah yang beroperasi dalam mode istimewa.



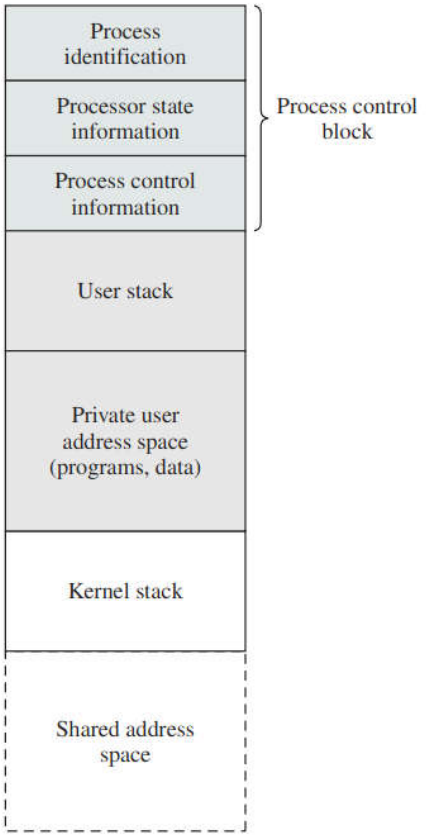
Gambar 3.1 Hubungan antara OS dan User Process

3.2. Execution within User Processes

Alternatif yang umum dengan sistem operasi pada komputer yang lebih kecil (PC, workstation) adalah menjalankan hampir semua perangkat lunak OS dalam konteks proses pengguna. Pandangannya adalah bahwa OS pada dasarnya adalah kumpulan rutinitas yang dipanggil oleh pengguna melakukan berbagai fungsi, dijalankan dalam lingkungan proses pengguna.

Diilustrasikan pada Gambar 3.1b. Pada titik tertentu, OS mengelola n proses image. Setiap image tidak hanya mencakup juga program, data, dan area tumpukan untuk program kernel.

Gambar 3.2 menunjukkan struktur gambar proses yang khas untuk strategi ini. Tumpukan kernel terpisah digunakan untuk mengelola panggilan / pengembalian saat proses dalam mode kernel. Kode dan data sistem operasi berada di ruang alamat bersama dan dibagikan ke semua proses pengguna.



Gambar 3.2. Process Image: Operating System Executes within User Space

3.3. Process-Based Operating System

Alternatif lain, yang diilustrasikan pada Gambar 3.1c, adalah mengimplementasikan OS sebagai kumpulan proses sistem. Seperti pada opsi lain, perangkat lunak yang merupakan bagian dari kernel dijalankan dalam mode kernel. Namun, dalam kasus ini, fungsi kernel utama adalah diatur sebagai proses terpisah. Sekali lagi, mungkin ada sejumlah kecil kode pengalihan proses yang dijalankan di luar proses apa pun.

4. Manajemen Proses (Unix Server IV)

UNIX System IV menggunakan fasilitas proses yang sederhana namun kuat terlihat oleh pengguna. UNIX mengikuti model Gambar 3.1b, di mana sebagian besar OS dijalankan dalam lingkungan proses pengguna. UNIX menggunakan dua kategori proses: proses sistem dan proses pengguna. Proses sistem berjalan dalam mode kernel dan menjalankan kode sistem operasi untuk menjalankan fungsi administratif dan system *housekeeping*, seperti alokasi memori dan pertukaran proses. Proses

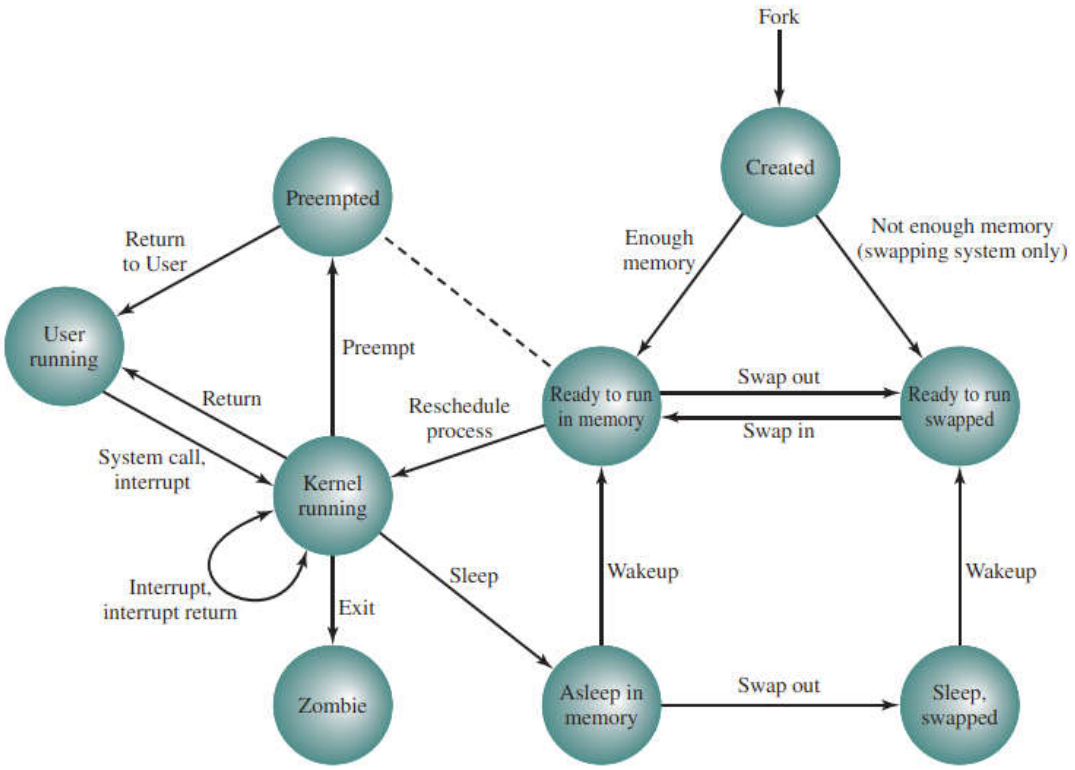
pengguna beroperasi dalam mode pengguna untuk menjalankan program dan utilitas pengguna dan dalam mode kernel untuk mengeksekusi instruksi yang dimiliki kernel. Proses pengguna memasuki mode kernel dengan mengeluarkan file panggilan sistem, ketika pengecualian (kesalahan) dihasilkan, atau ketika interupsi terjadi.

4.1. Proses State

Sebanyak sembilan status proses dikenali oleh sistem operasi UNIX SVR4;
ini tercantum dalam Tabel 6.1 dan diagram transisi keadaan ditunjukkan pada Gambar 6.1.

Tabel 6.1. UNIX Proses State

User Running	Executing in user mode.
Kernel Running	Executing in kernel mode.
Ready to Run, in Memory	Ready to run as soon as the kernel schedules it.
Asleep in Memory	Unable to execute until an event occurs; process is in main memory (a blocked state).
Ready to Run, Swapped	Process is ready to run, but the swapper must swap the process into main memory before the kernel can schedule it to execute.
Sleeping, Swapped	The process is awaiting an event and has been swapped to secondary storage (a blocked state).
Preempted	Process is returning from kernel to user mode, but the kernel preempts it and does a process switch to schedule another process.
Created	Process is newly created and not yet ready to run.
Zombie	Process no longer exists, but it leaves a record for its parent process to collect.



Gambar 6.1. UNIX Process Transition Diagram

Dua proses unik di UNIX. Proses 0 adalah proses khusus yang dibuat saat sistem melakukan boot; pada dasarnya, ini ditentukan sebelumnya sebagai struktur data yang dimuat pada saat boot. Ini adalah proses penukar (swapper). Selain itu, proses 0 memunculkan proses 1, disebut sebagai proses init; semua proses lain dalam sistem memiliki proses 1 sebagai leluhur/induk (ancestor). Ketika pengguna interaktif baru log on ke sistem, itu adalah proses 1 yang membuat proses pengguna untuk pengguna itu. Selanjutnya, proses pengguna dapat membuat anak proses dalam pohon percabangan, sehingga aplikasi tertentu dapat terdiri dari sejumlah proses terkait.

4.2. Process Description

Proses di UNIX adalah sekumpulan struktur data yang agak kompleks yang menyediakan OS dengan semua informasi yang diperlukan untuk mengelola dan mengirimkan proses. Tabel 6.2. merangkum elemen image proses, yang disusun menjadi tiga bagian: konteks tingkat pengguna, konteks daftar, dan konteks tingkat sistem.

Tabel 6.2. UNIX Process Image

User-Level Context	
Process text	Executable machine instructions of the program
Process data	Data accessible by the program of this process
User stack	Contains the arguments, local variables, and pointers for functions executing in user mode
Shared memory	Memory shared with other processes, used for interprocess communication
Register Context	
Program counter	Address of next instruction to be executed; may be in kernel or user memory space of this process
Processor status register	Contains the hardware status at the time of preemption; contents and format are hardware dependent
Stack pointer	Points to the top of the kernel or user stack, depending on the mode of operation at the time or preemption
General-purpose registers	Hardware dependent
System-Level Context	
Process table entry	Defines state of a process; this information is always accessible to the operating system
U (user) area	Process control information that needs to be accessed only in the context of the process
Per process region table	Defines the mapping from virtual to physical addresses; also contains a permission field that indicates the type of access allowed the process: read-only, read-write, or read-execute
Kernel stack	Contains the stack frame of kernel procedures as the process executes in kernel mode

Tabel 6.3. UNIX U Area

Process status	Current state of process.
Pointers	To U area and process memory area (text, data, stack).
Process size	Enables the operating system to know how much space to allocate the process.
User identifiers	The **real user ID** identifies the user who is responsible for the running process. The **effective user ID** may be used by a process to gain temporary privileges associated with a particular program; while that program is being executed as part of the process, the process operates with the effective user ID.
Process identifiers	ID of this process; ID of parent process. These are set up when the process enters thennCreated state during the fork system call.
Event descriptor	Valid when a process is in a sleeping state; when the event occurs, the process is transferred to a ready-to-run state.
Priority	Used for process scheduling.
Signal	Enumerates signals sent to a process but not yet handled.
Timers	Include process execution time, kernel resource utilization, and user-set timer used to send alarm signal to a process.
P_link	Pointer to the next link in the ready queue (valid if process is ready to execute).
Memory status	Indicates whether process image is in main memory or swapped out. If it is in memory, this field also indicates whether it may be swapped out or is temporarily locked into main memory

Konteks tingkat pengguna berisi elemen dasar dari program pengguna dan dapat dihasilkan langsung dari file objek yang dikompilasi. Program pengguna dipisahkan menjadi area teks dan data; area teks hanya-baca dan dimaksudkan untuk menampung instruksi program. Saat proses sedang dijalankan, prosesor menggunakan area tumpukan pengguna untuk pemanggilan dan pengembalian prosedur dan penerusan parameter. Yang dibagikan area memori adalah area data yang digunakan bersama dengan proses lain. Hanya ada satu salinan fisik dari area

memori bersama, tetapi, dengan menggunakan memori virtual, itu muncul ke setiap proses berbagi di mana wilayah memori bersama berada di ruang alamatnya. Kapan suatu proses tidak berjalan, informasi status prosesor disimpan di register area konteks.

Konteks tingkat sistem berisi informasi yang tersisa dari OS perlu mengelola prosesnya. Ini terdiri dari bagian statis, yang ukurannya tetap dan tetap dengan proses sepanjang masa, dan bagian dinamis, yang bervariasi ukuran melalui kehidupan proses. Salah satu elemen dari bagian statis adalah prosesnya entri tabel. Ini sebenarnya adalah bagian dari tabel proses yang dikelola oleh OS, dengan satu entri per proses. Entri tabel proses berisi informasi kontrol proses yang dapat diakses oleh kernel setiap saat; karenanya, dalam sistem memori virtual, semuanya entri tabel proses disimpan dalam memori utama. Tabel 6.2 mencantumkan isinya dari entri tabel proses. Area pengguna, atau area U, berisi informasi kontrol proses tambahan yang diperlukan oleh kernel saat dijalankan dalam konteks proses ini; itu juga digunakan saat proses paging ke dan dari memori. Tabel 6.3 menunjukkan isi tabel ini.

Perbedaan antara entri tabel proses dan area U mencerminkan fakta bahwa kernel UNIX selalu dijalankan dalam konteks beberapa proses. Banyak pada saat itu, kernel akan menangani masalah-masalah proses tersebut. Namun, terkadang, seperti saat kernel menjalankan algoritme penjadwalan persiapan untuk mengirimkan proses lain, itu akan membutuhkan akses ke informasi tentang proses lainnya. Informasi dalam tabel proses dapat diakses saat diberikan proses bukan yang sekarang.

Bagian statis ketiga dari konteks tingkat sistem adalah wilayah per proses tabel, yang digunakan oleh sistem manajemen memori. Terakhir, tumpukan kernel adalah bagian dinamis dari konteks tingkat sistem. Tumpukan ini digunakan saat proses sedang dijalankan dalam mode kernel dan berisi informasi yang harus disimpan dan dipulihkan saat panggilan prosedur dan interupsi terjadi.

4.3. Pengendalian proses

Proses pembuatan di UNIX dibuat melalui pemanggilan sistem kernel, fork( ). Kapan proses mengeluarkan permintaan fork, OS melakukan fungsi berikut:

1. Mengalokasikan slot dalam tabel proses untuk proses baru.
2. Memberikan ID proses unik untuk proses anak.

3. Membuat salinan dari gambar proses induk, dengan pengecualian dalam berbagi memori.
4. Menambah penghitung untuk setiap file yang dimiliki oleh induk, untuk mencerminkan bahwa proses tambahan sekarang juga memiliki file tersebut.
5. Menetapkan proses anak ke status Siap Jalankan (ready to run).
6. Mengembalikan nomor ID anak ke proses induk, dan nilai 0 untuk proses anak.

Semua pekerjaan ini diselesaikan dalam mode kernel dalam proses induk. Kapan kernel telah menyelesaikan fungsi-fungsi ini, ia dapat melakukan salah satu hal berikut, sebagai bagian dari rutinitas dispatcher:

- Tetap dalam proses induk. Kontrol kembali ke mode pengguna pada titik panggilan fork induk.
- Transfer kontrol ke proses anak. Proses anak mulai dijalankan pada titik yang sama dalam kode sebagai induk, yaitu di kembalian dari panggilan fork.
- Transfer kontrol ke proses lain. Baik induk dan anak ditinggalkan di Status Siap Jalankan.

Mungkin sulit untuk memvisualisasikan metode penciptaan proses ini karena keduanya baik induk dan anak menjalankan bagian kode yang sama. Perbedaannya adalah: Ketika pengembalian dari garpu terjadi, parameter pengembalian diuji. Jika nilainya nol, maka ini adalah proses anak, dan cabang dapat dijalankan sesuai kebutuhan program pengguna untuk melanjutkan eksekusi. Jika nilainya bukan nol, maka ini adalah induknya proses, dan jalur utama eksekusi dapat dilanjutkan.

RINGKASAN

Konsep paling mendasar dalam OS modern adalah prosesnya. Fungsi utama OS adalah membuat, mengelola, dan menghentikan proses. Saat \ prosesnya aktif, OS harus melihat bahwa masing-masing dialokasikan waktu untuk eksekusi oleh prosesor, mengoordinasikan aktivitas mereka, mengelola permintaan yang bertentangan, dan mengalokasikan sistem sumber daya untuk proses.

Untuk menjalankan fungsi manajemen prosesnya, OS menyimpan deskripsi dari setiap proses, atau image proses, yang mencakup ruang alamat di mana file proses dijalankan, dan blok kontrol proses. Yang terakhir berisi semua informasi yang diperlukan oleh OS untuk mengelola proses, termasuk statusnya saat ini, sumber daya yang dialokasikan untuknya, prioritas, dan data relevan lainnya.

Selama masa hidupnya, suatu proses bergerak di antara sejumlah kondisi. Yang paling yang terpenting adalah Siap, Berjalan, dan Diblokir. Proses yang siap adalah salah satunya saat ini tidak sedang dijalankan tetapi siap untuk dijalankan segera setelah OS mengirimkannya. Proses yang sedang berjalan adalah proses yang saat ini sedang dijalankan oleh prosesor. Dalam sistem multi-prosesor, lebih dari satu proses dapat dilakukan. Proses yang diblokir menunggu penyelesaian beberapa acara, seperti operasi I / O.

Sebuah proses yang sedang berjalan diinterupsi baik oleh interupsi, yang merupakan peristiwa itu terjadi di luar proses dan yang dikenali oleh prosesor, atau dengan mengeksekusi panggilan supervisor ke OS. Dalam kedua kasus, prosesor melakukan switch mode, mentransfer kontrol ke rutinitas sistem operasi. OS, setelah selesai pekerjaan yang diperlukan, dapat melanjutkan proses yang terputus atau beralih ke proses yang lain.

C. Daftar Pustaka

1. Operating System, Internals and design Principles, William Stallings 7th Ed. 2012
2. Modern Operating System 4th Ed. Andrew S Tanembaun 2009