

# **LAPORAN SISTEM OPERASI**

## **Praktikum 7**

### **“ Unix System Call dan Manajemen Memory”**



**FARID AZIZ WICAKSONO**

**1841720094**

**POLITEKNIK NEGERI MALANG**

**MEI 2019**

## Tugas Pendahuluan

Jawablah pertanyaan-pertanyaan berikut ini:

1. Apa yang dimaksud dengan system call?

Jawab: System call adalah tata cara pemanggilan di program aplikasi untuk memperoleh layanan yang disediakan oleh sistem operasi. System call berupa rutinitas sistem operasi untuk keperluan tertentu yang spesifik. System Call untuk manajemen proses diperlukan untuk mengatur proses-proses yang sedang berjalan.

2. Apa yang dimaksud dengan system call fork(), execl() dan wait(). Jawablah dengan menggunakan perintah man (contoh : man 2 fork, man 2 execl dan man 2 wait)?

Jawab:

- System call fork() digunakan untuk membuat proses. Tidak perlu ada argumen dan mengembalikan ID proses. Tujuan fork () adalah untuk menciptakan proses baru, yang menjadi proses anak dari pemanggil. Setelah proses anak baru dibuat, kedua proses akan menjalankan instruksi berikutnya.

```
FORK(2) Linux Programmer's Manual FORK(2)
NAME
    fork - create a child process
SYNOPSIS
    #include <unistd.h>

    pid_t fork(void);
DESCRIPTION
    fork() creates a new process by duplicating the calling
    process. The new process is referred to as the child
    process. The calling process is referred to as the par-
ent process.

    The child process and the parent process run in separate
    memory spaces. At the time of fork() both memory spaces
    have the same content. Memory writes, file mappings
    (mmap(2)), and unmappings (munmap(2)) performed by one
    of the processes do not affect the other.

    The child process is an exact duplicate of the parent
    process except for the following points:

    * The child has its own unique process ID, and this PID
      does not match the ID of any existing process group
      (setpgid(2)) or session.

    * The child's parent process ID is the same as the par-
      ent's process ID.

    * The child does not inherit its parent's memory locks
      (mlock(2), mlockall(2)).
```

- System call execl() adalah fungsi dari sistem operasi yang menjalankan file yang dapat dieksekusi dalam konteks proses yang sudah ada, menggantikan eksekusi sebelumnya. Tindakan ini juga disebut sebagai overlay.

```
EXEC(3) Linux Programmer's Manual EXEC(3)
NAME
    execl, execlp, execlx, execv, execvp, execvpe - execute
    a file
SYNOPSIS
    #include <unistd.h>

    extern char **environ;

    int execl(const char *path, const char *arg, ...
              /* (char *) NULL */);
    int execlp(const char *file, const char *arg, ...
              /* (char *) NULL */);
    int execlx(const char *path, const char *arg, ...
              /*, (char *) NULL, char * const envp[]
              */);
    int execv(const char *path, char *const argv[]);
    int execvp(const char *file, char *const argv[]);
    int execvpe(const char *file, char *const argv[],
               char *const envp[]);

    Feature Test Macro Requirements for glibc (see fea-
    ture_test_macros(7)):

    execvpe(): _GNU_SOURCE
DESCRIPTION
    The exec() family of functions replaces the current
    process image with a new process image. The functions
    described in this manual page are front-ends for
    execve(2). (See the manual page for execve(2) for fur-
    ther details about the replacement of the current
    process image.)
Manual page execl(3) line 1 (press h for help or q to quit)
```

- System call `wait()` adalah fungsi dari sistem operasi yang digunakan untuk memblokir proses pemanggilan sampai salah satu proses turunannya keluar atau sinyal diterima.

```

WAIT(2)                                Linux Programmer's Manual                                WAIT(2)
NAME
wait, waitpid, waitid - wait for process to change state
SYNOPSIS
#include <sys/types.h>
#include <sys/wait.h>

pid_t wait(int *wstatus);

pid_t waitpid(pid_t pid, int *wstatus, int options);

int waitid(idtype_t idtype, id_t id, siginfo_t *info,
int options); /* This is the glibc and POSIX inter-
               face; see NOTES for information on the raw sys-
               tem call. */

Feature Test Macro Requirements for glibc (see fea-
ture_test_macros(7)):

waitid():
    _XOPEN_SOURCE
    || /* Since glibc 2.12: */
    _POSIX_C_SOURCE >= 200809L
    || /* Glibc versions <= 2.19: */ _BSD_SOURCE
DESCRIPTION
All of these system calls are used to wait for state
changes in a child of the calling process, and obtain
information about the child whose state has changed. A
state change is considered to be: the child terminated;
the child was stopped by a signal; or the child was
Manual page wait(2) line 1 (press h for help or q to quit)

```

3. Apa yang dimaksud system virtual memory, proses swapping dan buffer cache pada manajemen memory?

Jawab:

- Virtual Memory adalah teknik manajemen memori yang dikembangkan untuk kernel multitugas. Teknik ini divirtualisasikan dalam berbagai bentuk arsitektur komputer di dalam penyimpanan data (seperti RAM dan Harddisk), yang memungkinkan sebuah program harus dirancang seolah-olah hanya ada satu jenis memori yang digunakan yaitu memori “virtual”, yang bertindak secara langsung pada alamat memori baca/tulis (RAM)
- Swapping adalah Suatu metode pengalihan proses yang bersifat sementara dari memori utama ke suatu tempat penyimpanan sementara (disk) dan dipanggil kembali ke memori jika akan melakukan eksekusi. Adapun proses pindahkan yaitu proses yang di blocked ke disk dan hanya memasukkan proses-proses ready ke memory utama.
- Buffer Cache adalah buffer cache adalah proses yang terjadi pada daerah memori yang menyimpan data ketika data sedang dipindahkan dari perangkat satu ke perangkat lainnya atau dari perangkat ke aplikasi yang memiliki akses data yang lebih cepat dan efisien.

4. Apa yang dimaksud perintah `free` dan `cat/proc/meminfo`?

Jawab:

- ❖ `Free` adalah perintah yang digunakan untuk melihat memori yang terpakai oleh suatu proses dan memperlihatkan swap space dan banyaknya swapping yang terjadi.
- ❖ `Cat /proc/meminfo` adalah perintah yang digunakan untuk melihat penggunaan memori dan swap-nya

5. Apa yang dimaksud perintah `ps`?

Jawab: `Ps` adalah perintah yang digunakan untuk melihat status proses yang terdiri dari attribut PID (Process Identifier), TTY, Time dan CMD

## Percobaan 1: Melihat proses parent dan proses child

No	Keterangan
1.	<p>Dengan menggunakan editor vi, buatlah file cpp dan ketikkan program berikut:</p> <pre>#include &lt;iostream&gt; using namespace std; #include &lt;sys/types.h&gt; #include &lt;unistd.h&gt; /* getpid() adalah system call yang dideklarasikan pada unistd.h    Menghasilkan suatu nilai dengan type pid_t.    pid_t adalah type khusus untuk process id yang ekuivalen dengan int */  int main(void){     pid_t mypid;     uid_t myuid;      for (int i = 0; i &lt; 3; i++){         mypid=getpid();         cout &lt;&lt; "I am process" &lt;&lt; mypid &lt;&lt; endl;         cout &lt;&lt; "My parent is process" &lt;&lt; getppid() &lt;&lt; endl;         cout &lt;&lt; "The owner of this process has uid" &lt;&lt; getuid() &lt;&lt; endl;         /* sleep adalah system call atau fungsi library yang menghentikan            proses ini dalam detik         */         sleep(1);     }     return 0; }  :wq</pre> <p>Analisa: listing program diatas berguna untuk menampilkan nomor PID dari proses dan parent serta user id dari user yang sedang aktif ketika program dieksekusi</p>
2.	<p>Gunakan g++ compiler untuk menjalankan program diatas</p> <pre>farid@farid-VirtualBox:~\$ g++ -o fork1 fork1.cpp farid@farid-VirtualBox:~\$ ./fork1 I am process 22451 My parent is process 2116 The owner of this process has uid 1000 I am process 22451 My parent is process 2116 The owner of this process has uid 1000 I am process 22451 My parent is process 2116 The owner of this process has uid 1000 farid@farid-VirtualBox:~\$</pre> <p>Analisa: perintah di atas digunakan untuk mengcompile file yang telah dibuat sebelumnya ke dalam bentuk file shell script sehingga dapat dieksekusi</p>
3.	<p>Amati output yang dihasilkan</p> <pre>I am process16938 My parent is process11725 The owner of this process has uid1000 I am process16938 My parent is process11725 The owner of this process has uid1000 I am process16938 My parent is process11725 The owner of this process has uid1000</pre> <p>Analisa: Hasil output menampilkan nomor PID dari proses, nomor PID dari parent dan user id dari user yang sedang aktif</p>

## Percobaan 2: Membuat dua proses terus menerus dengan sebuah system call fork()

No	Keterangan
1.	<p>Dengan menggunakan editor vi, buatlah file cpp dan ketikkan program berikut:</p> <pre>#include &lt;iostream&gt; using namespace std; #include &lt;sys/types.h&gt; #include &lt;unistd.h&gt;  /* getpid() dan fork() adalah system call yang dideklarasikan pada unistd.h. Menghasilkan suatu nilai dengan type pid_t. pid_t adalah type khusus untuk process id yang ekuivalen dengan int */  int main(void){     pid_t chldpid;     int x = 5;     chldpid = fork();     while (1) {         cout &lt;&lt; "This is process" &lt;&lt; getpid() &lt;&lt; endl;         cout &lt;&lt; "x is" &lt;&lt; x &lt;&lt; endl;         sleep(1);         x++;     }     return 0; }</pre> <p>Analisa: listing program diatas berguna untuk menampilkan nomor PID dari proses beserta urutan foreground-nya di dalam memory ketika program dieksekusi</p>
2.	<p>Gunakan g++ compiler untuk menjalankan program diatas. Pada saat dijalankan, program tidak akan pernah berhenti. Untuk menghentikan program tekan Ctrl+C.</p> <pre>farid@farid-VirtualBox:~\$ vi fork2.cpp farid@farid-VirtualBox:~\$ g++ -o fork2 fork2.cpp farid@farid-VirtualBox:~\$ ./fork2 This is process 22464 x is 5 This is process 22465 x is 5 This is process 22464 x is 6 This is process 22465 x is 6 This is process 22464 x is 7 This is process 22465 x is 7 This is process 22464 x is 8 This is process 22465 x is 8 This is process 22464 x is 9</pre> <p>Analisa: perintah di atas digunakan untuk mengcompile file yang telah dibuat sebelumnya ke dalam bentuk file shell script sehingga dapat dieksekusi</p>
3.	<p>Amati output yang dihasilkan</p> <pre>This is process16952 x is5 This is process16953 x is5 This is process16952 x is6 This is process16953 x is6 This is process16952 x is7 This is process16953 x is7 This is process16952 x is8 This is process16953 x is8 This is process16952</pre> <p>Analisa: Hasil output menampilkan nomor PID dari proses beserta urutan foreground-nya di dalam memory yang dilakukan secara berulang-ulang hingga user menghentikannya sendiri dengan perintah ctrl+c</p>

### Percobaan 3: Membuat dua proses sebanyak lima kali

No	Keterangan
1.	<p>Dengan menggunakan editor vi, buatlah file cpp dan ketikkan program berikut:</p> <pre>#include &lt;iostream&gt; using namespace std; #include &lt;sys/types.h&gt; #include &lt;unistd.h&gt;  /* getpid() dan fork() adalah system call yang dideklarasikan pada unistd.h. Menghasilkan suatu nilai dengan type pid_t. pid_t adalah type khusus untuk process id yang ekuivalen dengan int */  int main(void){     pid_t chilpid;     chilpid = fork();     for (int i = 0; i &lt; 5; i++){         cout &lt;&lt; "This is process" &lt;&lt; getpid() &lt;&lt; endl;         sleep(2);     }     return 0; }</pre> <p>Analisa: listing program diatas berguna untuk menampilkan nomor PID dari 2 buah proses yang dilakukan secara berulang sebanyak 5 kali perulangan.</p>
2.	<p>Gunakan g++ compiler untuk menjalankan program diatas</p> <pre>farid@farid-VirtualBox:~\$ g++ -o fork3 fork3.cpp farid@farid-VirtualBox:~\$ ./fork3 This is process 22480 This is process 22481 This is process 22480 This is process 22481 This is process 22480 This is process 22481 This is process 22480 This is process 22481 This is process 22480 This is process 22481 farid@farid-VirtualBox:~\$</pre> <p>Analisa: perintah di atas digunakan untuk mengcompile file yang telah dibuat sebelumnya ke dalam bentuk file shell script sehingga dapat dieksekusi</p>
3.	<p>Amati output yang dihasilkan</p> <pre>This is process16971 This is process16972 This is process16971 This is process16972 This is process16971 This is process16972 This is process16971 This is process16972 This is process16971 This is process16972</pre> <p>Analisa: Hasil output menampilkan nomor PID dari 2 buah proses yang dilakukan secara berulang sebanyak 5 kali perulangan</p>

## Percobaan 4: Proses parent menunggu sinyal dari proses child dengan system call wait

No	Keterangan
1.	<p>Dengan menggunakan editor vi, buatlah file cpp dan ketikkan program berikut:</p> <pre data-bbox="300 365 1252 685"> #include &lt;iostream&gt; using namespace std; #include &lt;sys/types.h&gt; #include &lt;unistd.h&gt; #include &lt;sys/wait.h&gt;  /* pid_t fork() dideklarasikan pada unistd.h.    pid_t adalah type khusus untuk process id yang ekuvalen dengan int */  int main(void){     pid_t child_pid;     int status;     pid_t wait_result;      child_pid = fork();     if (child_pid == 0){         /* kode ini hanya dieksekusi proses child */         cout &lt;&lt; "I am a child and my pid = " &lt;&lt; getpid() &lt;&lt; endl;         cout &lt;&lt; "My parent is" &lt;&lt; getpid() &lt;&lt; endl;         /* keluar jika akan menghentikan hanya proses child */     }     else if (child_pid &gt; 0){         /* kode ini hanya mengeksekusi proses parent */         cout &lt;&lt; "I am the parent and my pid = " &lt;&lt; getpid()             &lt;&lt; endl;          /* kode ini dieksekusi baik oleh proses parent dan child */         cout &lt;&lt; "I am a happy, healthy process and my pid = "             &lt;&lt; getpid() &lt;&lt; endl;         if (child_pid == 0){             /* kode ini hanya dieksekusi oleh proses child */             cout &lt;&lt; "I am a child and I am quitting work now!"                 &lt;&lt; endl;         }         else{             /* kode ini hanya dieksekusi oleh proses parent */             cout &lt;&lt; "I am a parent and I am going to wait for my child" &lt;&lt; endl;             do{                 /* parent menunggu sinyal SIGCHLD mengirinkan tanda                    bahwa proses child determinasi */                 wait_result = wait (&amp;status);             }while (wait_result != child_pid);             cout &lt;&lt; "I am a parent and I am quitting." &lt;&lt; endl;         }     }     return 0; } </pre> <p>Analisa: listing program di atas berguna untuk menampilkan nomor PID dari parent dan child dimana nomor PID ini nantinya akan dipanggil kembali pada pendeklarasian string berikutnya dan perulangan terjadi pada program ini sebanyak 2 perulangan dengan menampilkan nomor PID parent, nomor PID child, String parent dan String Child</p>
2.	<p>Gunakan g++ compiler untuk menjalankan program diatas</p> <pre data-bbox="300 981 1048 1279"> farid@farid-VirtualBox:~\$ g++ -o fork4 fork4.cpp farid@farid-VirtualBox:~\$ ./fork4 I am the parent and my pid = 17007 My child has pid = 17008 I am a happy, healthy process and my pid =17007 I am a parent and I am going to wait for my child I am a child and my pid = 17008 My parent is17007 I am a happy, healthy process and my pid =17008 I am a child and I am quitting work now! I am a parent and I am quitting. </pre> <p>Analisa: perintah di atas digunakan untuk mengcompile file yang telah dibuat sebelumnya ke dalam bentuk file shell script sehingga dapat dieksekusi</p>
3.	<p>Amati output yang dihasilkan</p> <pre data-bbox="300 1500 1114 1798"> I am the parent and my pid = 17007 My child has pid = 17008 I am a happy, healthy process and my pid =17007 I am a parent and I am going to wait for my child I am a child and my pid = 17008 My parent is17007 I am a happy, healthy process and my pid =17008 I am a child and I am quitting work now! I am a parent and I am quitting. </pre> <p>Analisa: Hasil output menampilkan nomor PID dari parent dan child dimana nomor PID ini nantinya akan dipanggil kembali pada pendeklarasian string berikutnya dan perulangan terjadi pada program ini sebanyak 2 perulangan dengan menampilkan nomor PID parent, nomor PID child, String parent dan String Child.</p>

**Percobaan 5: System call fork/exec dan wait mengeksekusi program bernama ls, menggunakan file executable /bin/ls dengan satu parameter -l yang ekuivalen dengan ls -l**

No	Keterangan
1.	<p>Dengan menggunakan editor vi, buatlah file cpp dan ketikkan program berikut:</p> <pre> #include &lt;iostream&gt; using namespace std; #include &lt;sys/types.h&gt; #include &lt;unistd.h&gt; #include &lt;sys/wait.h&gt;  /* pid_t fork() dideklarasikan pada unistd.h.    pid_t adalah type khusus untuk process id yang ekuivalen dengan int */  int main(void){     pid_t child_pid;     int status;     pid_t wait_result;      child_pid = fork();     if (child_pid == 0){         /* kode ini hanya dieksekusi proses child */         cout &lt;&lt; "I am a child and my pid = " &lt;&lt; getpid() &lt;&lt; endl;         execl("/bin/ls", "ls", "-l", "/home", NULL);         /* jika execl berhasil kode ini tidak pernah digunakan */         cout &lt;&lt; "Could not execl file/bin/ls" &lt;&lt; endl;         exit(1);         /* exit menghentikan hanya proses child */     }     else if(child_pid &gt; 0){         else if(child_pid &gt; 0){             /* kode ini hanya mengeksekusi proses parent */             cout &lt;&lt; "I am the parent and my pid = " &lt;&lt; getpid()                 &lt;&lt; endl;             cout &lt;&lt; "My child has pid = " &lt;&lt; child_pid &lt;&lt; endl;         }         else{             cout &lt;&lt; "The fork system call failed to create a new process" &lt;&lt; endl;             exit(1);         }         /* kode ini hanya dieksekusi oleh proses parent karena child            mengeksekusi dari "/bin/ls" atau keluar */         cout &lt;&lt; "I am a happy, healthy process and my pid = "             &lt;&lt; getpid() &lt;&lt; endl;         if(child_pid == 0){             /*kode ini tidak pernah dieksekusi */             printf("This code will never be executed!\n");         }         else{             /* kode ini hanya dieksekusi oleh proses parent */             cout &lt;&lt; "I am a parent and I am going to wait for my child" &lt;&lt; endl;             do{                 /* parent menunggu sinyal SIGCHLD mengirin tanda bila proses                    child determinasi */                 wait_result = wait(&amp;status);             }while(wait_result != child_pid);         }     }while(wait_result != child_pid);     cout &lt;&lt; "I am a parent and I am quitting." &lt;&lt; endl; } return 0; } </pre> <p>Analisa: listing program di atas berguna untuk menampilkan nomor PID dari parent dan child dimana PID tersebut nantinya akan dipanggil kembali pada pendeklarasian string berikutnya kemudian program menampilkan informasi tentang user yang sedang aktif dan mengakhiri eksekusi dengan tampilan “I am a parent and I am quitting.”</p>
2.	<p>Gunakan g++ compiler untuk menjalankan program diatas</p> <pre> farid@farid-VirtualBox:~\$ vi fork5.cpp farid@farid-VirtualBox:~\$ g++ -o fork5 fork5.cpp farid@farid-VirtualBox:~\$ ./fork5 I am the parent and my pid = 17052 My child has pid = 17053 I am a happy, healthy process and my pid = 17052 I am a parent and I am going to wait for my child I am a child and my pid = 17053 total 4 drwxr-xr-x 23 anggista anggista 4096 Mar 22 20:55 anggista I am a parent and I am quitting. </pre> <p>Analisa: perintah di atas digunakan untuk mengcompile file yang telah dibuat sebelumnya ke dalam bentuk file shell script sehingga dapat dieksekusi</p>
3.	<p>Amati output yang dihasilkan</p> <pre> I am the parent and my pid = 17052 My child has pid = 17053 I am a happy, healthy process and my pid = 17052 I am a parent and I am going to wait for my child I am a child and my pid = 17053 total 4 drwxr-xr-x 23 anggista anggista 4096 Mar 22 20:55 anggista I am a parent and I am quitting. </pre> <p>Analisa: Hasil output menampilkan nomor PID dari parent dan child dimana PID tersebut nantinya akan dipanggil kembali pada pendeklarasian string berikutnya kemudian program menampilkan informasi tentang user yang sedang aktif dan mengakhiri eksekusi dengan tampilan “I am a parent and I am quitting.”</p>



## Percobaan 6: System call fork/exec dan wait mengeksekusi program lain

No	Keterangan
1.	<p>Dengan menggunakan editor vi, buatlah file cpp dan ketikkan program berikut:</p> <pre>#include &lt;iostream&gt; using namespace std; #include &lt;sys/types.h&gt; #include &lt;unistd.h&gt; #include &lt;sys/wait.h&gt;  /* pid_t fork() dideklarasikan pada unistd.h.    pid_t adalah type khusus untuk process id yang ekuivalen dengan int    */  int main(void){     pid_t child_pid;     int status;     pid_t wait_result;      child_pid = fork();     if(child_pid == 0){         /* kode ini hanya dieksekusi proses child */         cout &lt;&lt; "I am a child and my pid = " &lt;&lt; getpid() &lt;&lt; endl;         execl ("fork3","goose", NULL);         /* jika execl berhasil kode ini tidak pernah digunakan */         cout &lt;&lt; "Could not execl file fork3" &lt;&lt; endl;         exit(1);         /* exit menghentikan hanya proses child */     }      cout &lt;&lt; "I am the parent and my pid = " &lt;&lt; getpid()         &lt;&lt; endl;     cout &lt;&lt; "My child has pid = " &lt;&lt; child_pid &lt;&lt; endl;     }     else {         cout &lt;&lt; "The fork system call failed to create a new process" &lt;&lt; endl;         exit(1);     }     /* kode ini hanya dieksekusi oleh proses parent karena child        mengeksekusi dari 'fork3' atau keluar */     cout &lt;&lt; "I am a happy, healthy process and my pid = "         &lt;&lt; getpid() &lt;&lt; endl;     if (child_pid == 0) {         /* kode ini tidak pernah dieksekusi */         printf("This code will never be executed!\n");     }     else{         /* kode ini hanya dieksekusi oleh proses parent */         cout &lt;&lt; "I am a parent and I am going to wait for my child" &lt;&lt; endl;         do{             /* parent menunggu sinyal SIGCHLD mengirtn tanda bila                proses child dterminasi */             wait_result = wait(&amp;status);         }while (wait_result != child_pid);         cout &lt;&lt; "I am a parent and I am quitting." &lt;&lt; endl;     }      return 0; }</pre> <p>Analisa: listing program di atas berguna untuk menampilkan nomor PID dari parent dan child dimana nomor PID tersebut nantinya akan dipanggil kembali pada pendeklarasian string berikutnya dan program juga menampilkan nomor PID dari 2 buah proses dan mengulanginya sebanyak 5 kali pengulangan</p>
2.	<p>Gunakan g++ compiler untuk menjalankan program diatas</p> <pre>farid@farid-VirtualBox:~\$ vi fork6.cpp farid@farid-VirtualBox:~\$ g++ -o fork6 fork6.cpp farid@farid-VirtualBox:~\$ ./fork6 I am the parent and my pid = 17099 My child has pid = 17100 I am a happy, healthy process and my pid = 17099 I am a parent and I am going to wait for my child I am a child and my pid = 17100 This is process17100 This is process17101 This is process17100 This is process17101 This is process17100 This is process17101 This is process17100 This is process17101 This is process17100 This is process17101 This is process17100 This is process17101 I am a parent and I am quitting.</pre> <p>Analisa: perintah di atas digunakan untuk mengcompile file yang telah dibuat sebelumnya ke dalam bentuk file shell script sehingga dapat dieksekusi</p>
3.	<p>Amati output yang dihasilkan</p> <pre>I am the parent and my pid = 17099 My child has pid = 17100 I am a happy, healthy process and my pid = 17099 I am a parent and I am going to wait for my child I am a child and my pid = 17100 This is process17100 This is process17101 This is process17100 This is process17101 This is process17100 This is process17101 This is process17100 This is process17101 This is process17100 This is process17101 This is process17100 This is process17101 I am a parent and I am quitting.</pre> <p>Analisa: Hasil output menampilkan nomor PID dari parent dan child dimana nomor PID tersebut nantinya akan dipanggil kembali pada pendeklarasian string berikutnya dan program juga menampilkan nomor PID dari 2 buah proses dan mengulanginya sebanyak 5 kali pengulangan</p>

## Percobaan 7: Melihat Manajemen Memory

No	Soal
1.	<p>Perhatikan dengan perintah dmesg jumlah memory tersedia dan proses swapping</p> <pre>farid@farid-VirtualBox:~\$ dmesg   more [    0.000000] Linux version 4.15.0-29-generic (buildd@lgw01-amd64-057) (gcc ve rsion 7.3.0 (Ubuntu 7.3.0-16ubuntu3)) #31-Ubuntu SMP Tue Jul 17 15:39:52 UTC 20 18 (Ubuntu 4.15.0-29.31-generic 4.15.18) [    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.15.0-29-generic root=UU ID=259de136-44e0-4b50-8c5c-80435872b98d ro quiet splash [    0.000000] KERNEL supported cpus: [    0.000000]   Intel GenuineIntel [    0.000000]   AMD AuthenticAMD [    0.000000]   Centaur CentaurHauls [    0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point reg isters'</pre> <p>Analisa: Pada gambar diatas terlihat bahwa jumlah memory yang tersedia adalah 154840 Kb dan jumlah memory swapping yang tersedia adalah 444072 Kb</p>
2.	<p>Dengan perintah free perhatikan jumlah memory "free", "used", "share" dan "buffer" .</p> <pre>farid@farid-VirtualBox:~\$ free               total        used         free       shared    buff/cache   available Mem:           1009124        711184         92972          4828        204968        155280 Swap:           483800          432792         51008</pre> <p>Analisa: Pada gambar di atas terlihat bahwa perintah free menyediakan layanan berupa menampilkan informasi mengenai penggunaan aktivitas memory yang terdiri dari total, used, free, shared, buffer/cache dan available</p>
3.	<p>Dengan perintah dibawah ini apakah hasilnya sama dengan no 2?</p> <pre>farid@farid-VirtualBox:~\$ cat /proc/meminfo MemTotal:        1009124 kB MemFree:          92848 kB MemAvailable:     155288 kB Buffers:          11636 kB Cached:           172888 kB SwapCached:       12648 kB Active:           425256 kB Inactive:         343828 kB Active(anon):     285064 kB Inactive(anon):   304344 kB Active(file):     140192 kB Inactive(file):   39484 kB Unevictable:      16 kB Mlocked:          16 kB SwapTotal:        483800 kB SwapFree:         51008 kB Dirty:            8 kB Writeback:         0 kB AnonPages:        572344 kB Mapped:           67292 kB Shmem:            4828 kB Slab:             53664 kB SReclaimable:     20576 kB SUnreclaim:       33088 kB KernelStack:      7920 kB</pre> <p>Analisa: Ya, hasilnya sama, namun pada perintah cat /proc/meminfo lebih banyak menampilkan informasi secara detail dan rinci mengenai aktivitas penggunaan memory</p>

4. Gunakan perintah dibawah ini

```
farid@farid-VirtualBox:~$ ls -lr /.
total 483908
lrwxrwxrwx  1 root root          30 Feb 12 21:32 vmlinuz -> boot/vmlinuz-4.15.0-29-generic
drwxr-xr-x 14 root root      4096 Jul 25  2018 var
drwxr-xr-x 10 root root      4096 Jul 25  2018 usr
drwxrwxrwt 14 root root      4096 Apr 24 14:08 tmp
dr-xr-xr-x 13 root root         0 Apr 24 08:12 sys
-rw-r----- 1 root root 495416320 Feb 12 21:23 swapfile
drwxr-xr-x  2 root root      4096 Jul 25  2018 srv
drwxr-xr-x 10 root root      4096 Feb 12 21:44 snap
drwxr-xr-x  2 root root     12288 Mar 13 08:26/sbin
drwxr-xr-x 26 root root       760 Apr 24 13:24 run
drwx----- 3 root root      4096 Jul 25  2018 root
dr-xr-xr-x 215 root root         0 Apr 24 08:13 proc
drwxr-xr-x  2 root root      4096 Jul 25  2018 opt
drwxr-xr-x  2 root root      4096 Jul 25  2018 mnt
drwxr-xr-x  2 root root      4096 Jul 25  2018 media
drwx----- 2 root root     16384 Feb 12 21:23 lost+found
drwxr-xr-x  2 root root      4096 Jul 25  2018 lib64
drwxr-xr-x 21 root root      4096 Feb 12 21:33 lib
lrwxrwxrwx  1 root root          33 Feb 12 21:23 initrd.img.old -> boot/initrd.img-4.15.0-29-generic
lrwxrwxrwx  1 root root          33 Feb 12 21:32 initrd.img -> boot/initrd.img-4.15.0-29-generic
```

Analisa: perintah diatas digunakan untuk menampilkan isi dari direktori root secara rinci

5. Perhatikan perubahan manajemen memory

```
farid@farid-VirtualBox:~$ free
              total        used         free       shared    buff/cache   available
Mem:          1009124        711676         91608          4828        205840        154808
Swap:          483800         432792         51008
```

Analisa: Perintah diatas digunakan untuk menampilkan aktivitas dari penggunaan memory dan swap secara general

6. Jalankan sebuah program, misalnya open Office. Perhatikan perubahan manajemen memory

```
farid@farid-VirtualBox:~$ free
              total        used         free       shared    buff/cache   available
Mem:          1009124        711704         90492          4828        206928        154772
Swap:          483800         432792         51008
```

Analisa: Pada saat kita menjalankan libre office maka memory akan menggunakan bagian pada dirinya yang masih free dan hal ini akan mengurangi jumlah total memory bebas yang tersedia, dimana sebelumnya (sebelum menjalankan libre office) memory yang tersedia sebanyak 145708 Kb dan setelah menjalankan libre office memory free berkurang menjadi 70424 Kb

7. Dengan perintah ps bagaimana penggunaan memory untuk setiap proses diatas?

```
farid@farid-VirtualBox:~$ ps -uax
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.4	372924	4756	?	Ss	09:48	0:05	/sbin/init spl
root	2	0.0	0.0	0	0	?	S	09:48	0:00	[kthreadd]
root	4	0.0	0.0	0	0	?	I<	09:48	0:00	[kworker/0:0H]
root	6	0.0	0.0	0	0	?	I<	09:48	0:00	[mm_percpu_wq]
root	7	0.0	0.0	0	0	?	S	09:48	0:00	[ksoftirqd/0]
root	8	0.0	0.0	0	0	?	I	09:48	0:02	[rcu_sched]
root	9	0.0	0.0	0	0	?	I	09:48	0:00	[rcu_bh]
root	10	0.0	0.0	0	0	?	S	09:48	0:00	[migration/0]
root	11	0.0	0.0	0	0	?	S	09:48	0:00	[watchdog/0]
root	12	0.0	0.0	0	0	?	S	09:48	0:00	[cpuhp/0]
root	13	0.0	0.0	0	0	?	S	09:48	0:00	[kdevtmpfs]
root	14	0.0	0.0	0	0	?	I<	09:48	0:00	[netns]
root	15	0.0	0.0	0	0	?	S	09:48	0:00	[rcu_tasks_kth
root	16	0.0	0.0	0	0	?	S	09:48	0:00	[kauditd]
root	17	0.0	0.0	0	0	?	S	09:48	0:00	[khungtaskd]
root	18	0.0	0.0	0	0	?	S	09:48	0:00	[oom_reaper]
root	19	0.0	0.0	0	0	?	I<	09:48	0:00	[writeback]
root	20	0.0	0.0	0	0	?	S	09:48	0:00	[kcompactd0]
root	21	0.0	0.0	0	0	?	SN	09:48	0:00	[ksmd]
root	22	0.0	0.0	0	0	?	SN	09:48	0:00	[khugepaged]
root	23	0.0	0.0	0	0	?	I<	09:48	0:00	[crypto]
root	24	0.0	0.0	0	0	?	I<	09:48	0:00	[kintegrityd]

Analisa: Dengan dijalankannya aplikasi libre office, tentunya penggunaan memory menjadi semakin besar dan menambah daftar proses yang terdaftar pada ps -uax. Melalui perintah ini kita bisa melihat berbagai proses yang sedang berjalan di memory, baik itu proses background maupun foreground secara detail. Pada gambar terlihat bahwa aplikasi libre office menggunakan sumber daya memory sebanyak 14.9% dan CPU 4.1%

## Latihan:

No	Soal
1.	<p>Ubahlah program fork5.cpp pada percobaan 5 untuk mengeksekusi perintah yang ekuivalen dengan</p> <ol style="list-style-type: none"> <li>ls -al /etc.</li> </ol> <pre> farid@farid-VirtualBox:~\$ ls -al /etc total 1116 drwxr-xr-x 124 root root 12288 Mar 13 08:55 . drwxr-xr-x 24 root root 4096 Feb 12 21:32 .. drwxr-xr-x 3 root root 4096 Jul 25 2018 acpi -rw-r--r-- 1 root root 3028 Jul 25 2018 adduser.conf -rw-r--r-- 1 root root 51 Feb 20 16:49 aliases -rw-r--r-- 1 root root 12288 Feb 20 16:49 aliases.db drwxr-xr-x 2 root root 4096 Feb 20 16:50 alternatives -rw-r--r-- 1 root root 401 Mei 29 2017 anacrontab -rw-r--r-- 1 root root 433 Okt 2 2017 app.conf drwxr-xr-x 6 root root 4096 Jul 25 2018 apm drwxr-xr-x 3 root root 4096 Mar 13 08:26 apparmor drwxr-xr-x 8 root root 4096 Mar 13 08:33 apparmor.d drwxr-xr-x 4 root root 4096 Jul 25 2018 apport -rw-r--r-- 1 root root 769 Apr 4 2018 appstream.conf drwxr-xr-x 6 root root 4096 Feb 12 21:37 apt drwxr-xr-x 3 root root 4096 Mar 13 08:29 avahi -rw-r--r-- 1 root root 2319 Apr 5 2018 bash.bashrc -rw-r--r-- 1 root root 45 Apr 2 2018 bash_completion drwxr-xr-x 2 root root 4096 Jul 25 2018 bash_completion.d -rw-r--r-- 1 root root 367 Jan 27 2016 bindresvport.blacklist drwxr-xr-x 2 root root 4096 Apr 20 2018 binfmt.d drwxr-xr-x 2 root root 4096 Jul 25 2018 bluetooth -rw-r----- 1 root root 33 Jul 25 2018 brlapi.key drwxr-xr-x 7 root root 4096 Jul 25 2018 brltty -rw-r--r-- 1 root root 25341 Apr 17 2018 brltty.conf </pre> <ol style="list-style-type: none"> <li>cat fork2</li> </ol> <pre> farid@farid-VirtualBox:~\$ cat fork2 ... Forking process 17155 ... </pre> <ol style="list-style-type: none"> <li>./fork2</li> </ol> <pre> x:~\$ ./fork2 This is process17155 x is5 This is process17156 x is5 This is process17155 x is6 This is process17156 x is6 This is process17155 x is7 This is process17156 x is7 This is process17155 x is8 This is process17156 x is8 </pre>

	<p>Analisa: Pada program diatas setelah listing program di buat maka program terlebih dahulu harus di compile ke dalam betuk shell script kemudian di eksekusi dan program menampilkan hasil output berupa semua isi (berserta file hidden) dari direktori /etc setelah itu program membuka fork2 dengan perintah cat dan kemudian program diakhiri dengan menampilkan 2 buah proses beserta PID nya yang dilakukan secara berulang-ulang</p>																					
2.	<p>Informasi apa saja mengenai manajemen memory yang ditampilkan pada perintah dmesg pada percobaan Anda?</p> <pre>farid@farid-VirtualBox:~\$ dmesg   more [    0.000000] Linux version 4.15.0-29-generic (buldd@lgw01-amd64-057) (gcc ve rsion 7.3.0 (Ubuntu 7.3.0-16ubuntu3)) #31-Ubuntu SMP Tue Jul 17 15:39:52 UTC 20 18 (Ubuntu 4.15.0-29.31-generic 4.15.18) [    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.15.0-29-generic root=UU ID=259de136-44e0-4b50-8c5c-80435872b98d ro quiet splash [    0.000000] KERNEL supported cpus: [    0.000000]   Intel GenuineIntel [    0.000000]   AMD AuthenticAMD [    0.000000]   Centaur CentaurHauls [    0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point reg isters' [    0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers' [    0.000000] x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers' [    0.000000] x86/fpu: xstate_offset[2]:  576, xstate_sizes[2]:  256 [    0.000000] x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using 'standard' format. [    0.000000] e820: BIOS-provided physical RAM map: [    0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable [    0.000000] BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved [    0.000000] BIOS-e820: [mem 0x0000000000f00000-0x0000000000ffffff] reserved</pre> <p>Analisa: Perintah diatas digunakan untuk menampilkan informasi berupa hardware yang sedang beraktivitas beserta file yang sedang dieksekusinya</p>																					
3.	<p>Bagaimana informasi yang ditampilkan dengan perintah free pada percobaan Anda?</p> <pre>farid@farid-VirtualBox:~\$ free</pre> <table><thead><tr><th></th><th>total</th><th>used</th><th>free</th><th>shared</th><th>buff/cache</th><th>available</th></tr></thead><tbody><tr><td>Mem:</td><td>1009124</td><td>712024</td><td>88992</td><td>4832</td><td>208108</td><td>154276</td></tr><tr><td>Swap:</td><td>483800</td><td>432280</td><td>51520</td><td></td><td></td><td></td></tr></tbody></table> <p>Analisa: Informasi yang ditampilkan pada percobaan saya adalah informasi berupa total memory dan swap, penggunaan memory dan swap, total memory dan swap yang sedang free, memory yang bisa dibagi, besaran buffering cache dan available memory</p>		total	used	free	shared	buff/cache	available	Mem:	1009124	712024	88992	4832	208108	154276	Swap:	483800	432280	51520			
	total	used	free	shared	buff/cache	available																
Mem:	1009124	712024	88992	4832	208108	154276																
Swap:	483800	432280	51520																			

4. Apa isi file /proc/meminfo pada percobaan yang Anda lakukan?

```
farid@farid-VirtualBox:~$ cat /proc/meminfo
MemTotal:      1009124 kB
MemFree:       116636 kB
MemAvailable:  173012 kB
Buffers:       10336 kB
Cached:        166816 kB
SwapCached:    10640 kB
Active:        387416 kB
Inactive:      355336 kB
Active(anon):  260484 kB
Inactive(anon): 309944 kB
Active(file):  126932 kB
Inactive(file): 45392 kB
Unevictable:   16 kB
Mlocked:       16 kB
SwapTotal:     483800 kB
SwapFree:      33684 kB
Dirty:         16 kB
Writeback:     0 kB
AnonPages:     555212 kB
Mapped:        57644 kB
Shmem:         4828 kB
Slab:          56236 kB
SReclaimable:  23148 kB
SUnreclaim:    33088 kB
KernelStack:   7916 kB
```

Analisa: Isi file /proc/meminfo adalah informasi berupa aktivitas penggunaan memory secara detail yang diantaranya berupa total memory, penggunaan memory, memory yang sedang free, besaran buffering dan cache dan available memory



5. Berapa besar memory yang digunakan setelah percobaan 7 dengan perintah ps – uax?

```
farid@farid-VirtualBox:~$ ps -uax
USER      PID  %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1   0.0   0.4 372924  4348 ?        Ss   09:48   0:05 /sbin/init spl
root         2   0.0   0.0      0     0 ?        S    09:48   0:00 [kthreadd]
root         4   0.0   0.0      0     0 ?        I<   09:48   0:00 [kworker/0:0H]
root         6   0.0   0.0      0     0 ?        I<   09:48   0:00 [mm_percpu_wq]
root         7   0.0   0.0      0     0 ?        S    09:48   0:00 [ksoftirqd/0]
root         8   0.0   0.0      0     0 ?        I    09:48   0:02 [rcu_sched]
root         9   0.0   0.0      0     0 ?        I    09:48   0:00 [rcu_bh]
root        10   0.0   0.0      0     0 ?        S    09:48   0:00 [migration/0]
root        11   0.0   0.0      0     0 ?        S    09:48   0:00 [watchdog/0]
root        12   0.0   0.0      0     0 ?        S    09:48   0:00 [cpuhp/0]
root        13   0.0   0.0      0     0 ?        S    09:48   0:00 [kdevtmpfs]
root        14   0.0   0.0      0     0 ?        I<   09:48   0:00 [netns]
root        15   0.0   0.0      0     0 ?        S    09:48   0:00 [rcu_tasks_kth
root        16   0.0   0.0      0     0 ?        S    09:48   0:00 [kauditd]
root        17   0.0   0.0      0     0 ?        S    09:48   0:00 [khungtaskd]
root        18   0.0   0.0      0     0 ?        S    09:48   0:00 [oom_reaper]
root        19   0.0   0.0      0     0 ?        I<   09:48   0:00 [writeback]
root        20   0.0   0.0      0     0 ?        S    09:48   0:00 [kcompactd0]
root        21   0.0   0.0      0     0 ?        SN   09:48   0:00 [ksmd]
root        22   0.0   0.0      0     0 ?        SN   09:48   0:00 [khugepaged]
root        23   0.0   0.0      0     0 ?        I<   09:48   0:00 [crypto]
root        24   0.0   0.0      0     0 ?        I<   09:48   0:00 [kintegrityd]
root        25   0.0   0.0      0     0 ?        I<   09:48   0:00 [kblockd]
root        26   0.0   0.0      0     0 ?        I<   09:48   0:00 [ata_sff]
root        27   0.0   0.0      0     0 ?        I<   09:48   0:00 [md]
```

Analisa: besarnya memory yang digunakan setelah percobaan 7 adalah 303648 Kb dengan penggunaan CPU sebesar 20.2%

6. Lakukan hal yang sama dengan percobaan 7 untuk melihat perubahan memory setelah dilakukan beberapa proses pada shell. Tentukan perintah yang dilakukan misalnya membuka browser dan perhatikan hal-hal berikut:

1. Informasi apa saja yang ditampilkan dengan perintah free?

```
farid@farid-VirtualBox:~$ free
              total        used        free      shared  buff/cache   available
Mem:      1009124        692380       116380         4828       200364       172820
Swap:      483800         450116        33684
```

Analisa: Informasi yang ditampilkan pada percobaan saya adalah informasi berupa total memory dan swap, penggunaan memory dan swap, total memory dan swap yang sedang free, memory yang bisa dibagi, besaran buffering cache dan available memory

2. Informasi apa saja yang disimpan file /proc/meminfo?



```

farid@farid-VirtualBox:~$ cat /proc/meminfo
MemTotal: 1009124 kB
MemFree: 116124 kB
Thunderbird Mail
MemAvailable: 173060 kB
Buffers: 10672 kB
Cached: 167040 kB
SwapCached: 10636 kB
Active: 387940 kB
Inactive: 355216 kB
Active(anon): 260336 kB
Inactive(anon): 309936 kB
Active(file): 127604 kB
Inactive(file): 45280 kB
Unevictable: 16 kB
Mlocked: 16 kB
SwapTotal: 483800 kB
SwapFree: 33684 kB
Dirty: 0 kB
Writeback: 0 kB
AnonPages: 555068 kB
Mapped: 57780 kB
Shmem: 4828 kB
Slab: 56292 kB
SReclaimable: 23148 kB
SUnreclaim: 33144 kB
KernelStack: 7916 kB
PageTables: 40580 kB
NFS_Unstable: 0 kB
Bounce: 0 kB

Shmem: 4828 kB
Slab: 56292 kB
SReclaimable: 23148 kB
SUnreclaim: 33144 kB
KernelStack: 7916 kB
PageTables: 40580 kB
NFS_Unstable: 0 kB
Bounce: 0 kB
WritebackTmp: 0 kB
CommitLimit: 988360 kB
Committed_AS: 4383776 kB
VmallocTotal: 34359738367 kB
VmallocUsed: 0 kB
VmallocChunk: 0 kB
HardwareCorrupted: 0 kB
AnonHugePages: 0 kB
ShmemHugePages: 0 kB
ShmemPmdMapped: 0 kB
CmaTotal: 0 kB
CmaFree: 0 kB
HugePages_Total: 0
HugePages_Free: 0
HugePages_Rsvd: 0
HugePages_Surp: 0
Hugepagesize: 2048 kB
DirectMap4k: 114624 kB
DirectMap2M: 933888 kB

```

Analisa: Isi file /proc/meminfo adalah informasi berupa aktivitas penggunaan memory secara detail yang diantaranya berupa total memory, penggunaan memory, memory yang sedang free, besaran buffering dan cache dan available memory

3. Berapa besar kapasitas memory total?

```
farid@farid-VirtualBox:~$ cat /proc/meminfo
MemTotal:      1009124 kB
```

4. Berapa kapasitas memory yang sudah terpakai?

```
farid@farid-VirtualBox:~$ free
              total        used
Mem:      1009124        692380
```

5. Berapa kapasitas memory yang belum terpakai?

```
farid@farid-VirtualBox:~$ free
              total        used         free
Mem:      1009124        692380        116380
```

6. Berapa kapasitas memory yang digunakan sharing beberapa proses?

```
farid@farid-VirtualBox:~$ free
              total        used         free       shared
Mem:      1009124        692380        116380         4828
```

7. Berapa kapasitas buffer cache?

```
farid@farid-VirtualBox:~$ free
              total        used         free       shared  buff/cache
Mem:      1009124        692380        116380         4828        200364
```

## Kesimpulan

Kesimpulan yang saya dapat setelah mengikuti praktikum ini dengan baik adalah:

- ❖ System call adalah tata cara pemanggilan di program aplikasi untuk memperoleh layanan yang disediakan oleh sistem operasi. System call berupa rutinitas sistem operasi untuk keperluan tertentu yang spesifik. System Call untuk manajemen proses diperlukan untuk mengatur proses-proses yang sedang berjalan.
- ❖ Manajemen memory adalah proses mengelola memori computer dengan efisien. Kebutuhan utama manajemen memory adalah untuk menyediakan cara secara dinamis dalam mengalokasikan bagian-bagian dari memori untuk program atas request yang dilakukan dan membebaskan memori untuk digunakan kembali ketika tidak lagi diperlukan.