7

Alokasi Memori

Modul 7

Alokasi Memori

7.1. Tujuan Praktikum

Materi praktikum pada bab ini bertujuan untuk memperkenalkan beberapa metode alokasi memori dari permasalahan yang ada di *contiguous memory allocation,* seperti solusi first-fit, best-fit, dan worst-fit terhadap ukuran proses-proses yang sedang berjalan.

7.2. Capaian Praktikum

1. Mahasiswa mampu memahami dan mempraktekkan metode alokasi memori seperti solusi first-fit, best-fit, dan worst-fit.

7.3. Dasar Teori

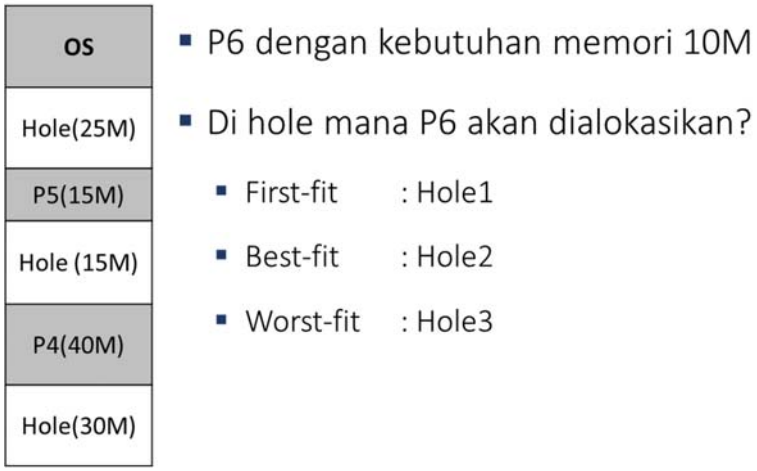
Secara umum, blok memori yang tersedia terdiri dari sekumpulan lubang (*hole*) dengan berbagai ukuran yang tersebar di seluruh memori. Ketika sebuah proses tiba dan membutuhkan memori, sistem mencari lubang yang cukup besar untuk proses ini. Jika lubangnya terlalu besar, itu dibagi menjadi dua bagian. Satu bagian dialokasikan untuk proses kedatangan; yang lain dikembalikan ke lubang. Ketika sebuah proses dihentikan, ia melepaskan blok memorinya, yang kemudian ditempatkan kembali di sekumpulan lubang.

Prosedur ini adalah contoh khusus dari masalah alokasi penyimpanan dinamis umum, yang menyangkut bagaimana memenuhi permintaan ukuran n dari daftar lubang yang bebas. Ada banyak solusi untuk masalah ini diantaranya first-fit, best-fit, dan worst-fit. Ketiganya merupakan teknik yang paling umum digunakan untuk memilih lubang bebas dari sekumpulan lubang yang tersedia. Berikut penjelasan ketiga teknik alokasi memori tersebut:

a. **First-fit**. Alokasikan lubang pertama yang cukup besar. Pencarian dapat dimulai baik di awal set lubang atau di lokasi di mana pencarian pertama sebelumnya berakhir. Kita bisa berhenti mencari begitu kita menemukan lubang bebas yang cukup besar.

b. **Best-fit**. Alokasikan lubang terkecil yang cukup besar. Kita harus mencari seluruh daftar lubang yang tersedia, kecuali daftar lubang itu diurutkan berdasarkan ukuran. Teknik ini menghasilkan lubang sisa terkecil.

c. **Worst-fit**. Alokasikan lubang terbesar. Sekali lagi, kita harus mencari seluruh daftar lubang, kecuali jika diurutkan berdasarkan ukuran. Teknik ini menghasilkan lubang sisa terbesar, yang mungkin lebih berguna daripada lubang sisa yang lebih kecil dari pendekatan best-fit.



7.4. Langkah Praktikum

Praktikum ini dilakukan dengan terlebih dahulu terhubung dengan layanan aws educate dengan cara mengaktifkan instance dari halaman instance summary. Pilih action dan Start untuk mengaktifkan instance. Lakukan koneksi SSH dengan cara yang sama seperti pada Bab 1. Lakukan percobaan pada kode program berikut.

7.4.1. Solusi first-fit

```
#include<stdio.h>
//mendefinisikan nilai variabel array maksimum sejumlah 20
#define max 20
void main()
{
    //set setiap variabel yang dibutuhkan
    int frag[max],ukuranBlok[max],ukuranProses[max],i,j,
        jmlBlok,jmlProses,temp;
    static int alokasiBlok[max],flags[max];
    //input dari pengguna
    printf("\nTeknik Alokasi Memori - First Fit");
    printf("\nMasukkan jumlah blok memori:");
    scanf("%d",&jmlBlok);
    printf("Masukkan jumlah proses:");
    scanf("%d",&jmlProses);
```

```
16      printf("\nMasukkan ukuran blok memori:-\n");
17      for(i=1;i<=jmlBlok;i++)
18      {
19          printf("Blok %d:",i);
20          scanf("%d",&ukuranBlok[i]);
21      }
22      printf("Masukkan ukuran proses :-\n");
23      for(i=1;i<=jmlProses;i++)
24      {
25          printf("Proses %d:",i);
26          scanf("%d",&ukuranProses[i]);
27      }
28      //mengambil setiap proses satu persatu
29      for(i=1;i<=jmlProses;i++)
30      {
31          for(j=1;j<=jmlBlok;j++)
32          {
33              //melakukan proses alokasi dengan mengurangi
34                ukuran blok dengan ukuran proses
35              if(alokasiBlok[j]!=1)
36              {
37                  temp=ukuranBlok[j]-ukuranProses[i];
38                  if(temp>=0)
39                  {
40                      flags[i]=j;
41                      break;
42                  }
43              }
44          }
45          //alokasi first-fit berdasarkan informasi alokasi blok memori
46          frag[i]=temp;
47          alokasiBlok[flags[i]]=1;
48      }
49     //tampilan output
50      printf("\nNo. Proses:\tUkuran Proses :\tAlokasi Blok:\t
51             Ukuran Blok:\tFragment");
52      for(i=1;i<=jmlProses;i++)
53      printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,ukuranProses[i],flags[i],
54             ukuranBlok[flags[i]],frag[i]);
55  }
```

a. Simpan berkas tersebut dengan nama alloc-first-fit.c

b. Kompilasi program alloc-first-fit.c melalui terminal dengan menuliskan perintah

gcc alloc-first-fit.c –o alloc-first-fit

c. Jalankan program dengan menuliskan perintah **.\alloc-first-fit.exe**

d. Masukkan variable berikut:

e. Jumlah proses 5, jumlah proses 4. Ukuran blok secara berurutan 15, 20, 5, 7, 4. Ukuran proses secara berurutan 2, 5, 6, 14.

f. Capture outputnya dan simpan sebagai laporan.

g. Amati output hasil percobaanya untuk referensi pada pembahasan.

7.4.2. Solusi best-fit

```c
#include<stdio.h>
//mendefinisikan nilai variabel array maksimum sejumlah 20
#define max 20
void main()
{
    //set setiap variabel yang dibutuhkan
    int frag[max],ukuranBlok[max],ukuranProses[max],i,j,
        jmlBlok,jmlProses,temp,lowest=10000;
    static int alokasiBlok[max],flags[max];
    //input dari pengguna
    printf("\nTeknik Alokasi Memori - Best Fit");
    printf("\nMasukkan jumlah blok memori:");
    scanf("%d",&jmlBlok);
    printf("Masukkan jumlah proses:");
    scanf("%d",&jmlProses);
    printf("\nMasukkan ukuran blok memori:-\n");
    for(i=1;i<=jmlBlok;i++)
    {
        printf("Blok %d:",i);
        scanf("%d",&ukuranBlok[i]);
    }
    printf("Masukkan ukuran proses :-\n");
    for(i=1;i<=jmlProses;i++)
    {
        printf("Proses %d:",i);
        scanf("%d",&ukuranProses[i]);
    }
    for(i=1;i<=jmlProses;i++)
    {
        for(j=1;j<=jmlBlok;j++)
        {
            //melakukan proses alokasi dengan mengurangi
```

```
33              ukuran blok dengan ukuran proses
34            if(alokasiBlok[j]!=1)
35            {
36                temp=ukuranBlok[j]-ukuranProses[i];
37               if(temp>=0)
38                    if(lowest>temp)
39                    {
40                        flags[i]=j;
41                        lowest=temp;
42                    }
43            }
44        }
45        //alokasi best-fit berdasarkan nilai fragment terkecil
46        frag[i]=lowest;
47        alokasiBlok[flags[i]]=1;
48        lowest=10000;
49    }
50   //tampilan output
51    printf("\nNo. Proses:\tUkuran Proses :\tAlokasi Blok:\t
52           Ukuran Blok:\tFragment");
53    for(i=1;i<=jmlProses;i++)
54    printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,ukuranProses[i],flags[i],
55           ukuranBlok[flags[i]],frag[i]);
56}
```

a. Simpan berkas tersebut dengan nama alloc-best-fit.c

b. Kompilasi program alloc-best-fit.c melalui terminal dengan menuliskan perintah

c. gcc alloc-best-fit.c –o alloc-best-fit

d. Jalankan program dengan menuliskan perintah **.\alloc-best-fit.exe**

e. Masukkan variable berikut:

f. Jumlah proses 5, jumlah proses 4. Ukuran blok secara berurutan 15, 20, 5, 7, 4. Ukuran proses secara berurutan 2, 5, 6, 14.

g. Capture outputnya dan simpan sebagai laporan.

h. Amati output hasil percobaanya untuk referensi pada pembahasan.

7.4.3. Solusi worst-fit

```c
#include<stdio.h>
//mendefinisikan nilai variabel array maksimum sejumlah 20
#define max 20
void main()
{
    //set setiap variabel yang dibutuhkan
    int frag[max],ukuranBlok[max],ukuranProses[max],i,j,
        jmlBlok,jmlProses,temp,highest=0;
    static int alokasiBlok[max],flags[max];
    //input dari pengguna
    printf("\nTeknik Alokasi Memori - Worst Fit");
    printf("\nMasukkan jumlah blok memori:");
    scanf("%d",&jmlBlok);
    printf("Masukkan jumlah proses:");
    scanf("%d",&jmlProses);
    printf("\nMasukkan ukuran blok memori:-\n");
    for(i=1;i<=jmlBlok;i++)
    {
        printf("Blok %d:",i);
        scanf("%d",&ukuranBlok[i]);
    }
    printf("Masukkan ukuran proses :-\n");
    for(i=1;i<=jmlProses;i++)
    {
        printf("Proses %d:",i);
        scanf("%d",&ukuranProses[i]);
    }
    for(i=1;i<=jmlProses;i++)
    {
        for(j=1;j<=jmlBlok;j++)
        {
            //melakukan proses alokasi dengan mengurangi
              ukuran blok dengan ukuran proses
            if(alokasiBlok[j]!=1)
            {
                temp=ukuranBlok[j]-ukuranProses[i];
               if(temp>=0)
                    if(highest<temp)
                    {
                        flags[i]=j;
                        highest=temp;
                    }
            }
```

```
        }
        //alokasi worst-fit berdasarkan nilai fragment terbesar
        frag[i]=highest;
        alokasiBlok[flags[i]]=1;
        highest=0;
    }
   //tampilan output
    printf("\nNo. Proses:\tUkuran Proses :\tAlokasi Blok:\t
           Ukuran Blok:\tFragment");
    for(i=1;i<=jmlProses;i++)
    printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,ukuranProses[i],flags[i],
           ukuranBlok[flags[i]],frag[i]);
}
```

a. Simpan berkas tersebut dengan nama alloc-worst-fit.c

b. Kompilasi program alloc-best-fit.c melalui terminal dengan menuliskan perintah

c. gcc alloc-worst-fit.c –o alloc-worst-fit

d. Jalankan program dengan menuliskan perintah **.\alloc-worst-fit.exe**

e. Masukkan variable berikut:

f. Jumlah proses 5, jumlah proses 4. Ukuran blok secara berurutan 15, 20, 5, 7, 4. Ukuran proses secara berurutan 2, 5, 6, 14.

g. Capture outputnya dan simpan sebagai laporan.

h. Amati output hasil percobaanya untuk referensi pada pembahasan.

7.5. Pembahasan

1. Berdasarkan hasil pengamatan Anda pada output setiap berkas first-fit, best-fit, dan worst-fit, maka:
   a. Bagaimana hasil alokasi blok memori untuk setiap variable yang telah dimasukkan?
   b. Apakah ada proses yang tidak teralokasikan pada blok memori tertentu? Jika ada, berikan penjelasan!
2. Jelaskan bagian kode program yang membuat teknik alokasi memori ini berbeda! Ulas untuk setiap teknik alokasi memori!

3. Jelaskan fragment yang dimaksud pada tampilan output program tersebut! Apa pengaruhnya terhadap blok memori?

7.6. Kesimpulan

Jelaskan kesimpulan dari hasil percobaan pada Bab 7 ini

UNTUK DIPERHATIKAN

Setiap selesai melakukan praktikum, jangan lupa menonaktfikan instance agar billing kuota tidak terus berjalan. Lakukan dengan cara memilih button Action pada Instance Summary. Pilih Stop sebagai perintah yang harus dijalankan.