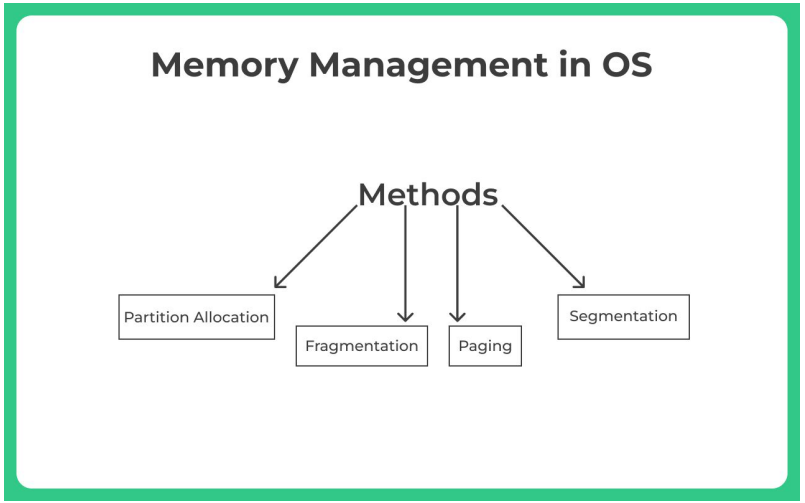
Bellek Yönetimi Projesi

Muhammed Salih Yılmaz
21 Ağustos 2023

Nedir?

Projenin amacı, İşletim Sistemlerinde arka planda gerçekleşen bellek yönetiminin simüle edilmesidir.

Belleğe OS tarafından yapılan allocation ve deallocation işlemleri, linked list mantığı kullanılarak görselleştirilecektir.



Neler yapılacak?

- **Bellek İadesi:**
  - Kullanıcının verdiği adres, eğer belli koşulları sağlıyorsa, free edilmiş gibi kullanıcıya boş olarak verilecektir. Bu boş alanlar bir single linked-list'te tutulacak ve kullanıcıya gösterilecektir.
- **Bellek Alma, verilen adres ile**
  - Kullanıcı, daha önce boşalmış olan adreslerden yer isteyebilir. İstenilen adres, belli kurallar içerisinde değilse kullanıcıya tahsis edilmez. Kullanıcı bellek alma işlemini doğrudan adres belirterek gerçekleştirir.
- **Bellek Alma, best-fit ile**
  - Benzer şekilde burada da kullanıcı, istediği boyuttaki belleği ister. Fakat doğrudan adresi belirtmez. Bunun yerine yazılan simülasyon, istenilen boyuttaki alanı boş olan adreslerden best-fit algoritmasına göre bulur ve tahsis eder.

Kullanıcı program ile nasıl etkileşime geçecek?

Simülasyonda kullanıcıya 3 adet interface çıkılmıştır. Programı kullanan kişi, arka planda neler gerçekleştiğini bilmeden bu interface'leri (bu durumda fonksiyonları) kullanır ve bellek iadesi ve bellek tahsisini yapar.

```
void bellekIadeEt    (struct dugum** root, int startAddr, int size);

void bellekAlAdresli (struct dugum** root, int startAddr, int size);

void bellekAlAdressiz(struct dugum** root, int size);
```

Kullanıcı ne görecek?

Önceden de belirtildiği üzere, kullanıcının gördüğü alanlar **sadece iade edilmiş (yani boş gözüken) alanlardır.** Bu alanlar bir single linked-list ile implement edilmiştir.

Kullanıcı bellekteki boş alanları, ekrandaki print'ler vasıtasıyla görür. Bir dosyaya yazılmaz.

```
struct dugum {
        int bas;
        int son;
        struct dugum * next;
};
```

BELLEK İADESİ

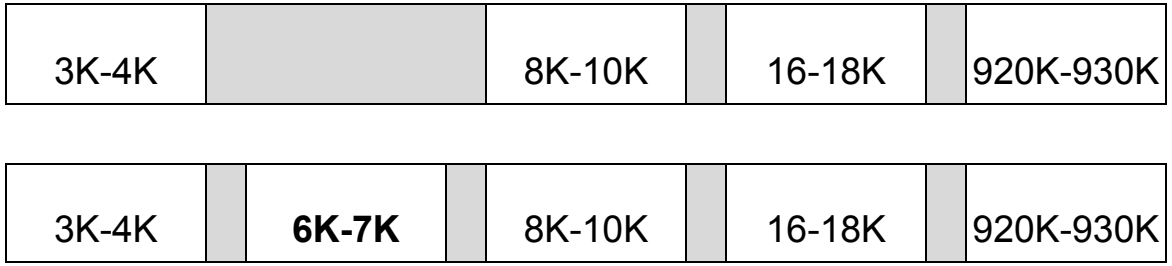
Bellek iadesi nasıl gerçekleşecek?

1. Tamamı doluysa -> iade edilecek
2. Bir kısmı dolu bir kısmı boşsa -> hiçbir şey yapmayacak.
3. Tamamı zaten boşsa -> hiçbir şey yapmayacak.
4. Adres, aralığa geliyorsa veya bitişikse -> birleştirme yapacak.

Tamamı dolu durum
İade

1. İstenilen adresten itibaren tamamını iade edecek.

1 İade 6K-7K



Bir kısmı dolu, bir kısmı boş durum İade

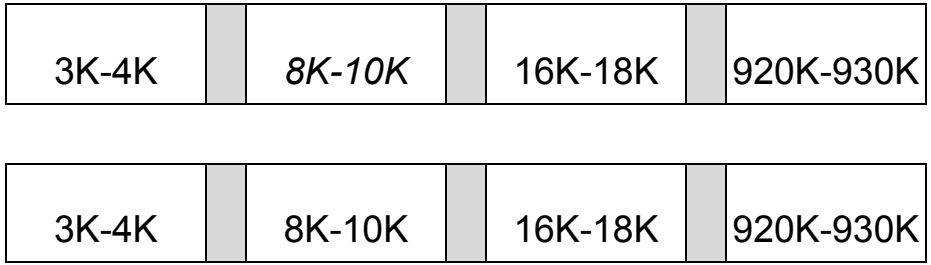
2. Hiçbir şey yapmayacak.

		9K-11K	3K-4K		*8K-10K*		16K-18K		920K-930K
2	**İade**	**7K-9K**							
		7K-11K	3K-4K		8K-10K		16K-18K		920K-930K

Tamamı boş durum
İade

3. Hiçbir şey yapmayacak.

3 İade 8K-10K

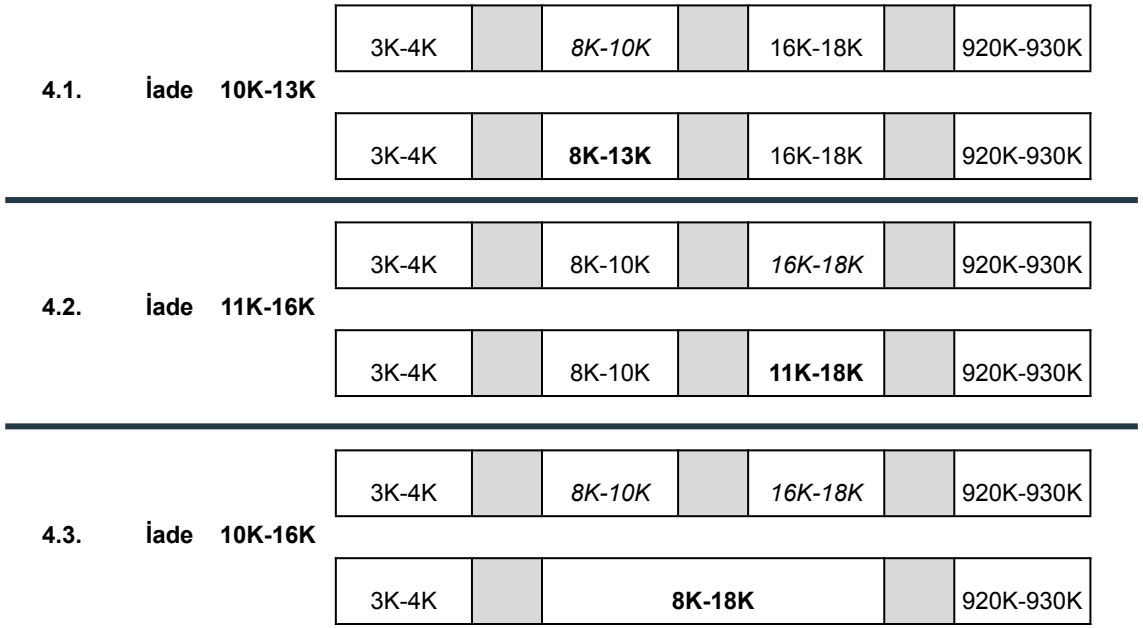


Birleştirme durumları
İade

4.1. Sonundan birleştirme

4.2. Başından birleştirme

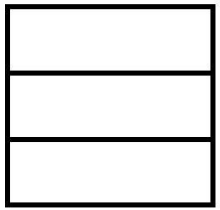
4.3. Ortadan birleştirme



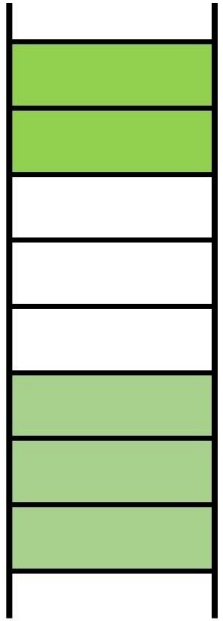
ADRESLİ BELLEK ALMA

Bellek Alma durumları Adresli

1. Girilen adres boş alanın başından önce; bitişik değil
2. Girilen adres boş alanın başından önce; bitişik
3. Girilen adres ile boş alan birebir aynı
4. Girilen adres boş alanın içinde; baştan bitişik
5. Girilen adres boş alanın içinde; başa ve sona değmiyor
6. Girilen adres boş alanın içinde; sonundan bitişik
7. Girilen adres boş alanın sonundan sonra; bitişik
8. Girilen adres boş alanın sonundan sonra; bitişik değil
9. Girilen adres boş alanın başından başlıyor sonundan büyük. Yani bir kısmı dolu bir kısmı boş
10. Girilen adres boş alanın öncesinden başlıyor sonuna eşit. Yani bir kısmı dolu bir kısmı boş



Process



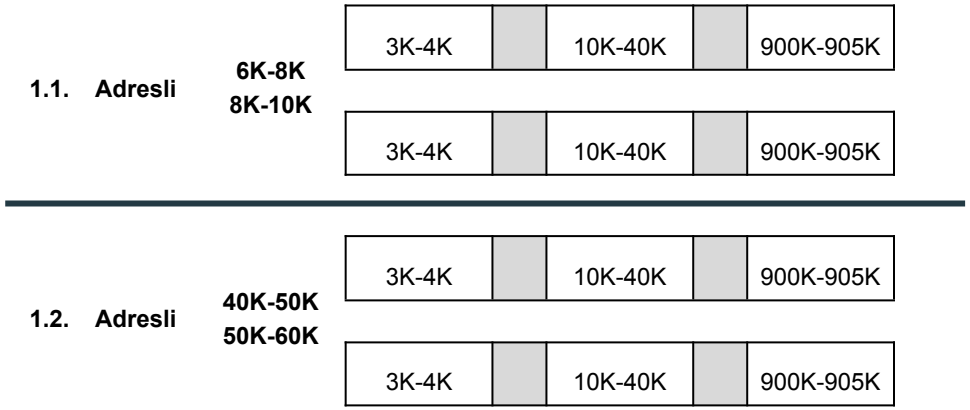
Memory blocks

Dolu adresten bellek alma

Bu durumda hiçbir şey yapmayacak.

1.1. Adres node’un öncesinde

1.2. Adres node’un sonrasında



Boş adresten bellek alma

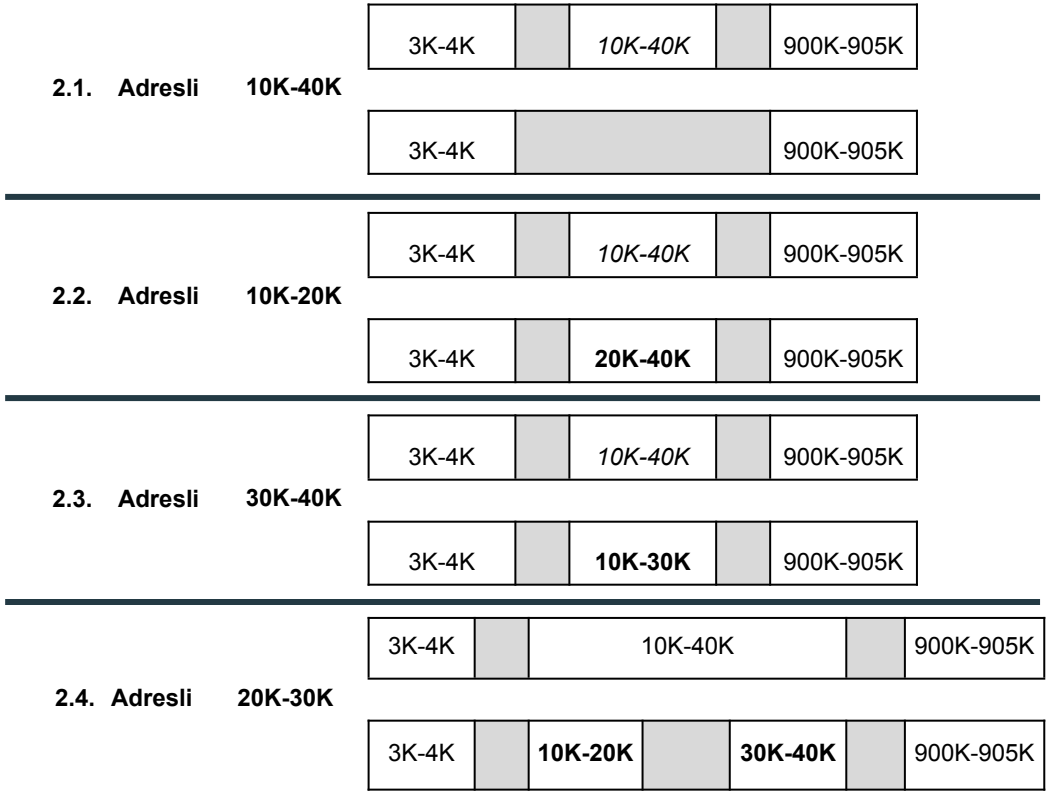
Bellek alınacak, node güncellenecek.

2.1. Girilen adres ile boş alan birebir aynı
    2.1.1. Tek node olma durumu

2.2. Baştan bitişik

2.3. Sondan bitişik

2.4. Tamamen içinde

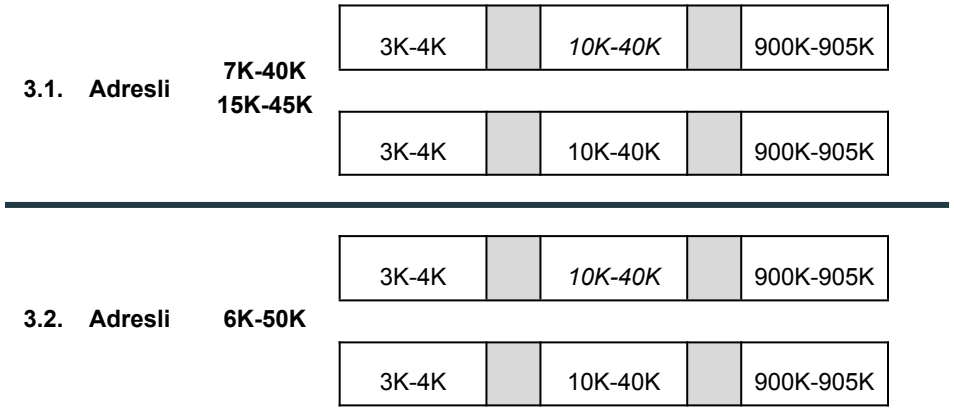


Bir kısmı dolu bir kısmı boş adresten bellek alma

Bu durumda hiçbir şey yapmayacak.

3.1. Adresin bir kısmı dolu bir kısmı boş

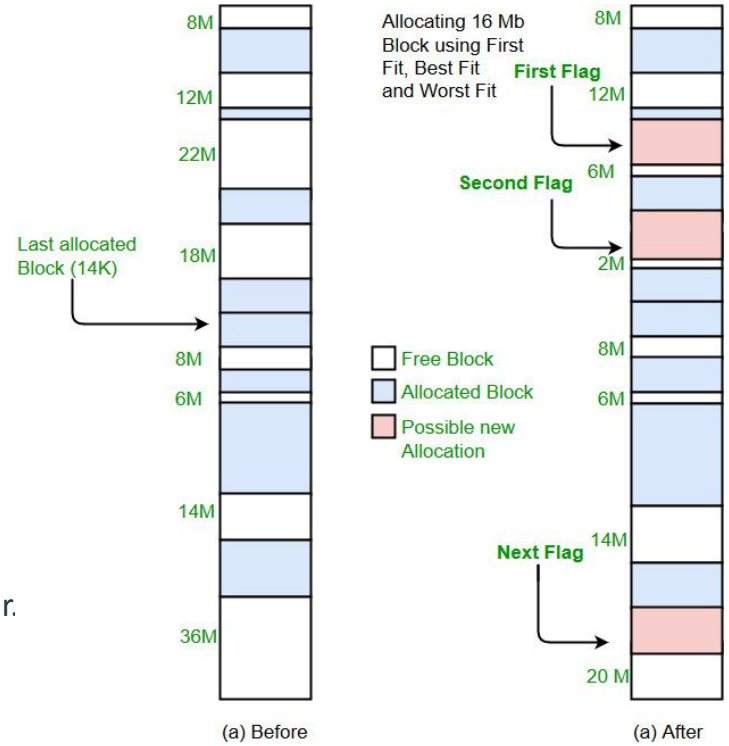
3.2. Adres boş-dolu-boş şeklinde



ADRESSİZ BELLEK ALMA

Bellek Alma durumları Adressiz

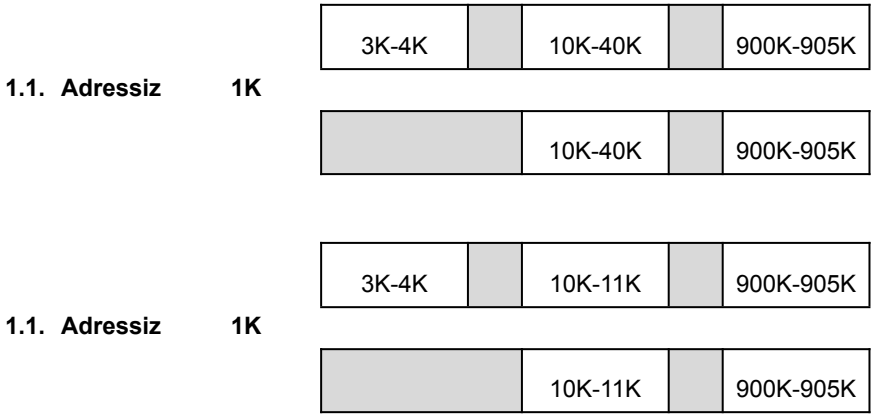
1. Listenin tamamı taranır.
2. İstenen alan ile boş alanın boyut farkı hesaplanır.
3. Birebir uyan alan best-fit'tir.
   a. Birden fazla birebir uyan varsa, ilki kabul edilir.
4. Değilse en küçük boş alan best-fit olarak döndürülür.
   a. Birden fazla aynı en küçük alan varsa, ilki kabul edilir.
5. Best-fit bulunan alan girilen boyut ile doldurulur.
6. Doldurma işlemi node'un başlangıcından itibaren yapılır.
   a. Uygulama gereksinimleri bunu istemektedir



Birebir uygunluk durumu
Adressiz bellek alma

1.1. Birebir uyuyor, tek uygun

1.2. Birebir uyuyor, birden çok, ilkini alır



En küçük fark durumu
Adressiz bellek alma

2.1. En ufak fark tek yerde

2.2. En ufak fark birden çok yerde, ilkini alır