# LiNUX BELLEK KULLANIMIN ÖğRENiLMESi



## Özet

Bu dokümanla Linux'ta RAM,SWAP alanının kullanımı ve erişim şekilleri .Ayrıca /proc dizini hakkında bilgi ve örnekler verilecektir .Bu döküm anı anlamanız için C bilginiz olması konuyu daha iyi anlamanızı sağlayabilir.

Bu doküman giriş niteliğindedir. Kodlarda grafik Ara birimi,kanal programlama. gibi kodu büyütecek unsurlardan kaçınılmıştır.

## Proc Dizini

Proc Dizini linux sistemi çalıştığında linux kernel'i tarafından oluşturulan sistem bilgilerinin tutulduğu bir dizindir. Bu dizinde CPU tipinden,bellek kullanımına ,disk yapısı,proses bilgileri ve birçok bilgiye ulaşabiliriz. Örneğin en basit bellek kullanımın görmek istiyorsanız yapmanız gereken

**# cat /proc/meminfo**

derseniz sistem size şöyle bir çıktı verecektir.(Sizin bilgisayarınızda RAM oranına göre değişebilir.) :

| | total: | used: | free: | shared: | buffers: | cached: |
|---|---|---|---|---|---|---|
| Mem: | 294531072 | 266133504 | 28397568 | 0 | 66404352 | 85819392 |
| Swap: | 271425536 | 0 | 271425536 | | | |
| MemTotal: | 287628 | KB | | | | |
| MemFree: | 27732 | KB | | | | |
| MemShared: | 0 | KB | | | | |
| Buffers: | 64848 | KB | | | | |
| Cached: | 83808 | KB | | | | |
| SwapCached: | 0 | KB | | | | |
| Active: | 189288 | KB | | | | |
| ActiveAnon: | 70744 | KB | | | | |
| ActiveCache: | 118544 | KB | | | | |
| Inact_dirty: | 25056 | KB | | | | |
| Inact_laundry: | 0 | KB | | | | |
| Inact_clean: | 5804 | KB | | | | |
| Inact_target | 44028 | KB | | | | |
| HighTotal: | 0 | KB | | | | |
| HighFree: | 0 | KB | | | | |
| LowTotal: | 287628 | KB | | | | |
| LowFree: | 27732 | KB | | | | |
| SwapTotal: | 265064 | KB | | | | |
| SwapFree: | 265064 | KB | | | | |

Şeklinde bir çıktı olacaktır .Bu şekilde cpuinfo(cpu tipi ve özellikleri),harddisk bilgileriniz vb.. sistem bilgileri /proc dizinden inceleyebilirsiniz.

## Bellek Kullanımın Öğrenim Çeşitleri

Linux'te bellek kullanımının öğrenilmesi ile ilgili değişik yöntemler bulunmaktadır. Ben Kullandığım 3 değişik yöntemi bu dokümanda anlatmaktayım.

**1.Yöntem :** En basit olan yöntem ***/proc/meminfo*** dosyasını açıp programımızda kullanmak. Bu istediğiniz her hangi bir dille bu yöntemi uygulayabilirsiniz.

**2.Yöntem :** Linux sürümlerinde bulabileceğiniz veya İnternet'ten indirebileceğiniz bu kütüphane veya yazılım geliştirme ara birimi ile olan **Glibtop** kütüphanesi. Bu

kütüphane ile CPU kullanımından, Kullanılan RAM alanında dahil birçok kütüphanesi olan güzel ve kullanışlı bir kütüphane.

**örnek kullanımı :**

**systeminfo.c**

```
/*
iskender atasoy
email iskenderatasoy@yahoo.co.uk
free usage
*/
#include <stdio>
#include <glibtop.h>
#include <glibtop/cpu.h>//cpu hakkında bilgi alabileceğimiz header
#include <glibtop/mem.h>//bellek hakkında bilgi alabileceğimiz header
#include <glibtop/proclist.h>//prosesler hakkında bilgi alabileceğimiz header

int main(){
//CPU hakkında bilgilerin depolanacağı struct yapısı
glibtop_cpu cpu;
//Bellek kullanımı hakkında bilgilerin depolanacağı struct yapısı
glibtop_mem memory;
//Proses kullanımı hakkında bilgilerin depolanacağı struct yapısı
glibtop_proclist proclist;


//mutlaka glibtop_init fonksiyonunu çağırıp sistemi
//hazırlamalıyız.
glibtop_init();

//CPU hakkında bilgiler alınıyor
glibtop_get_cpu (&cpu);

//ekrana aktarılıyor
printf("CPU TYPE INFORMATIONS \n\n"
      "Cpu   Total     : %ld \n"
      "Cpu   User      : %ld \n"
      "Cpu   Nice      : %ld \n"
      "Cpu   Sys       : %ld \n"
      "Cpu   Idle      : %ld \n"
      "Cpu   Frequences : %ld \n",
         (unsigned long)cpu.total,
         (unsigned long)cpu.user,
         (unsigned long)cpu.nice,
         (unsigned long)cpu.sys,
         (unsigned long)cpu.idle,
         (unsigned long)cpu.frequency);
```

```
//bellek kullanımı
glibtop_get_mem(&memory);

printf("\nMEMORY USING\n\n"
      "Memory   Total      : %ld MB\n"
      "Memory   Used       : %ld MB\n"
      "Memory   Free       : %ld MB\n"
      "Memory   Buffered   : %ld MB\n"
      "Memory   Cached     : %ld MB\n"
      "Memory   user       : %ld MB\n"
      "Memory   Locked     : %ld MB\n",
         (unsigned long)memory.total/(1024*1024),//Toplam RAM
         (unsigned long)memory.used/(1024*1024),//Kullanılan RAM
         (unsigned long)memory.free/(1024*1024),//Boş RAM miktarı
         (unsigned long)memory.shared/(1024*1024),//paylaşılan ram
         (unsigned long)memory.buffer/(1024*1024),//tampon bellek alanı
         (unsigned long)memory.cached/(1024*1024),//cache alanın boyutu
         (unsigned long)memory.user/(1024*1024),//kullanıcı alanı
         (unsigned long)memory.locked/(1024*1024));//özel alanların boyutu

int which,arg;
//istenilen proses hakkında bilgi alınır
glibtop_get_proclist(&proclist,which,arg);
printf("%ld\n%ld\n%ld\n",
        (unsigned long)proclist.number,//proses nosu
        (unsigned long)proclist.total,//proses toplamı
        (unsigned long)proclist.size);//proses boyu
return 0;
}
```

**Makefile**

```
CC=gcc
CFLAGS=-Wall -g
CLIBS=-lgtop-2.0 -lgtop_sysdeps-2.0 -lgtop_common-2.0

cpuinfo:cpu.c
      $(CC) $(CFLAGS) systeminfo.c -o systeminfo  $(CLIBS)
clean:
      rm -f systeminfo
```

Çalıştırmak için sadece **make** demeniz yeter. Programı silmek için **make clean** deminiz. Programı **./systeminfo** ile çalıştırabilirsiniz. Hata olarak glibtop bulunamadı der ise kütüphaneyi kendiniz kurmanız gerekecektir.

3.Yöntem : Linux Kernel'inden faydalanmaktır .Linux'te Bellek kullanımı ile ilgili kullanılan **sysinfo(..)** kütüphanesiyle de bellek oranın tespiti mümkündür. Konsoldan

**#man sysinfo** bize sysinfo ilgili dokümanı(Man İngilizcedeki Manuals sözcüğünün kısaltmasıdır. Türkçe tercümesi el kitabı, kılavuz(bkz. konsolda **man man** derseniz detaylı kullanımını görürsünüz.)) çıktı olarak konsola

## SYNOPSIS

#include <sys/sysinfo.h>
int sysinfo(struct sysinfo *info);

## DESCRIPTION

Until Linux **2.3.16**, sysinfo used to return information in the following
structure:

struct sysinfo {
long uptime; /* Seconds since boot */
unsigned long loads[3]; /* 1, 5, and 15 minute load averages */
unsigned long totalram; /* Total usable main memory size */
unsigned long freeram; /* Available memory size */
unsigned long sharedram; /* Amount of shared memory */
unsigned long bufferram; /* Memory used by buffers */
unsigned long totalswap; /* Total swap space size */
unsigned long freeswap; /* swap space still available */
unsigned short procs; /* Number of current processes */
char _f[22]; /* Pads structure to 64 bytes */
};
and the sizes were given in bytes. Since Linux **2.3.23** (i386),**2.3.48**
(all architectures) the structure is

struct sysinfo {
long uptime; /* Seconds since boot */
unsigned long loads[3]; /* 1, 5, and 15 minute load averages */
unsigned long totalram; /* Total usable main memory size */
unsigned long freeram; /* Available memory size */
unsigned long sharedram; /* Amount of shared memory */
unsigned long bufferram; /* Memory used by buffers */
unsigned long totalswap; /* Total swap space size */
unsigned long freeswap; /* swap space still available */
unsigned short procs; /* Number of current processes */
unsigned long totalhigh; /* Total high memory size */
unsigned long freehigh; /* Available high memory size */
unsigned int mem_unit; /* Memory unit size in bytes */
char _f[20-2*sizeof(long)-sizeof(int)]; /* Padding for libc5 */
};

## RETURN VALUE

On success, zero is returned. On error, -1 is returned, and errno is set appropriately.
bize fonksiyonun hangi kütüphanede olduğu hakkında bilgi vermektedir. Burada

önemli noktalardan biride iki farklı struct yapısının olmasıdır. Yukarıda yardım metninde **2.3.16** kernel versiyonuna kadar **1. struct** yapısı **2.3.23** den **2.3.48** arasında **2. struct** yapısı kullanılmaktadır diyor. Ayrıca hata olduğunda ise -1 geri dönmekte imiş. Hangi kernel versiyonu olduğunu konsoldan uname -a veya cat /proc/version diyerek öğrenebilirsiniz. Şimdi Biraz parmaklarımı çalıştırma zamanı .Yapmak istediğimiz şey basit biz çıkmak isteyene kadar sistem sürekli/belli aralıklarda sistem bilgisini bize versin. Örneğimiz için şimdilik sadece RAM ve SWAP bilgisi versin.

### systeminfo2.c

```
/*
iskender atasoy
email iskenderatasoy@yahoo.co.uk

free usage
*/
#include <stdio.h>        //printf fonksiyonu
#include <stdlib.h>       //ekranı silmek için system() fonksiyonu
#include <string.h>       //strerror(..) ve strcpy için
#include <errno.h>        //errno değişkeni için
#include <unistd.h>       //sleep fonksiyonu için
#include <sys/sysinfo.h>  //sysinfo(..) fonksiyonu
#include <linux/version.h> //sistem kernel'in numarası için

//Funksiyon Prototipimiz
void get_ram_info();

int main(){
   get_ram_info();
return 0;
}

void get_ram_info()
{
  struct sysinfo system_info;   //RAM ve SWAP hakkında bilgi alacağımız yapı
  int error;                    //hata kodumuz
  while(1){                     //Biz kapatana kadar çalışsın
  error=sysinfo(&system_info);  //gerekli bilgileri al
  if(error < 0){//hata varmı ? Her zaman kontrol etmemiz gerek

  error=errno;
  //errno *nix sistemler için global hata denetleyicisidir.
  //yani hata numarası buna yazılır.Ama bazı durumlarda
  //kernel bloke edebilir.En güvenilir yol geçici bir nesneye atamaktır.
  //hata numarasının okunabilir forma dönüştürmesi
  /* Error Found ...Print an error message and exit. */
  fprintf (stderr, "Error Defination is: %s\n", strerror (error));
  exit (1);     //programdan çık
```

```
 }

 //linuxte en kolay ekranı silme komutudur.
 //Burada clear bash komutudur.
 system("clear");

//Ayrıca Linux'ta Bir konsol prog. CTRL+C ile sonlandıra bilirsiniz.
printf("\t\t\tMemory Browser Text Versions\n");
printf("\t\t\tTo Exit use CTR+C\n\n");
printf("\t\t\tTotal      Ram Size      : %ld\n",(long)system_info.totalram);
printf("\t\t\tFree       Ram Size      : %ld\n",(long)system_info.freeram);
printf("\t\t\tShared     Ram Size      : %ld\n",(long)system_info.sharedram);
printf("\t\t\tBuffered   Ram Sizi      : %ld\n",(long)system_info.bufferram);
printf("\t\t\tTotal Swap Size         : %ld\n",(long)system_info.totalswap);
printf("\t\t\tFree  Swap Size         : %ld\n",(long)system_info.freeswap);
printf("\t\t\tNumber Of Proses        : %d \n",(short int)system_info.procs);
  //these code block works 2.3.23-48 kernel version
  //if our kernel number less than it error can happend
  //if kernel >2.3.23 use these else not
  //Evet buraya kadar sıradan bir koddu fakat
  //kernel nosunu alıp iki struct yapısı içinde çalışabilir hale getirdik.
  //Böylece programımızın çaşılabilirliğini 2.3.48 nolu sürüme kadar garantiledik.
  if(strcmp("2.3.23",UTS_RELEASE)<0 && strcmp(UTS_RELEASE,"2.3.48")>0){
   printf("\t\t\tTotal High Memory Size  : %ld\n",(long)system_info.totalhigh);
   printf("\t\t\tAvailable High Mem.Size : %ld\n",(long)system_info.freehigh);
   printf("\t\t\tMemory Unit Sizes       : %ld\n",(long)system_info.mem_unit);
  }
 //Sleep komutu *UNIX sistemlerinde belirtilen süre kadar sistem bekler.
 //bizim örneğimiz 2 sn bekliyecek yoksa ekranda çok hızlı bir geçiş olacaktı.
 sleep(2);
 }
}
```

Derlemek için
**gcc -Wall systeminfo2.c -o systeminfo2**
demeniz yeter.

**Son Söz**

Eğer buraya kadar tahammül ettiyseniz :)) linux bellek tespiti için 3 yöntemi göstermiş olmuş bulunmaktayız. Her Türlü görüş ve öneriyi mail adresime ilete bilirsiniz. mail adresim iskenderatasoy@yahoo.co.uk