|||.Systemcallimplementation

Brk()systemcallimplementation

Implementingthebrk()systemcallinvolvesmodifyingtheendofaprocess’sdata segment(i.e.,theheap).Here'sahigh-leveloverviewfollowedbyabasicexampleofhow thiscouldbedoneinaLinux-likekernelenvironment(forlearningorOSdevelopment purposes).

Overviewofbrk()

Purpose:Setstheendofthedatasegment(heap)toaspecifiedvalue.

Signature:intbrk(void*end_data_segment);

Ifsuccessful,itreturns0;onerror,itreturns-1andsetserrno.

Kernel-LevelPseudocodeforbrk() Assume:

currentisthecurrentprocessdescriptor.

Theprocesshasamm_structthatholdsmemorysegmentinfo.

Ccode

```
SYSCALL_DEFINE1(brk,unsignedlong,end_data_segment)
{
   structmm_struct*mm=current->mm; unsignedlongold_brk,new_brk;

   down_write(&mm->mmap_sem);//lock

   old_brk=mm->brk; new_brk=PAGE_ALIGN(end_data_segment);

   if(new_brk<mm->start_brk||new_brk>mm->end_code){ up_write(&mm
   ->mmap_sem);//unlock return-ENOMEM;
   }
```

```
    if(new_brk>old_brk){
            if(expand_heap(mm,old_brk,new_brk)<0){
        up_write(&mm->mmap_sem); return-ENOMEM;
      }
    }

    mm->brk=new_brk; up_write(&mm->mmap_sem);

    return0;
}
```

User-SpaceExample(forunderstanding) ccode

```
#include<unistd.h> #include<stdio.h>

intmain(){
    void*initial_brk=sbrk(0); printf("Initialbrk:%p\n",initial_brk);

    void*new_brk=(char*)initial_brk+4096; if(brk(new_brk)==0){
      printf("brkmovedto:%p\n",new_brk);
    }else{
      perror("brkfailed");
    }

    return0;
}
```

howtosimulatethisinaminimalkernel,orimplementacustomversioninatoyOS

Goal:Simulatebrk()inaToyKernel We'llassume:

Asimplememorymanagerisalreadyinplace.

Yourkernelhasbasicsyscallhandling.

Theheapisacontiguoussegmentthatcangrowupward.

Step1:DefineHeapRegioninYourKernel

Inyourprocessstruct(e.g.,task_struct),defineheapbounds:

ccode

```
structtask_struct{ ...
    void*heap_start; void*heap_end;//currentendofheap(brk)
    void*heap_max;//maximumallowed
};
```

Duringprocesscreation:

ccode

```
process->heap_start=alloc_some_memory(); process->heap_end=process->heap_start;
process->heap_max=process->heap_start+HEAP_LIMIT; Step2:Addthesys_brkSystemCall
c
```

Copycode

```
intsys_brk(void*new_end){
    structtask_struct*current=get_current_task();

    if(new_end==0){
            return(int)current->heap_end;//returncurrentbreak
    }
```

```
    if((char*)new_end<(char*)current->heap_start||
       (char*)new_end>(char*)current->heap_max){ return-1;//invalidrange
    }

    current->heap_end=new_end; return(int)new_end;
}
```

Step3:Hooksys_brkintoYourSyscallTable Addasyscallnumberforbrk,like:

ccode

```
#defineSYS_BRK45
```

Inyoursyscalldispatcher:

ccode

```
switch(syscall_num){ ...
    caseSYS_BRK:
        ret=sys_brk((void*)arg0); break;
}
```

Step4:Testbrk()inUserland(ifyousimulateuserprograms) ccode

```
intmain(){
    void*current=syscall(SYS_BRK,0); printf("Heapstart:%p\n",current);

    void*new_end=current+0x1000; syscall(SYS_BRK,new_end);

    void*check=syscall(SYS_BRK,0); printf("Heapnow:%p\n",check);
}
```

Thisisasimplifiedview,

Ref.

Google

TutorialsPoint – brk() System Call in Unix

YouTube – sbrk/brk System Calls and Optimistic Allocation Explained