САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО КАФЕДРА КОМПЬЮТЕРНЫХ СИСТЕМ И ПРОГРАММНЫХ ТЕХНОЛОГИЙ

Отчёт по лабораторной работе Nol Курс: <Проектирование ОС и компонентов>

Тема: <Системные вызовы>

Вьнолнил студент:

Бояркин Никита Сер,ее ич Грунна: 13541/3

Про ерил:

Душутина Елена Владимиро на

Санкт-Петербур, 2018 ,.

# Содержание

[l Лабораторная работа Nol](#_bookmark0) 2

* 1. [Цель работь](#_bookmark1) 2
  2. [Про,рамма работь](#_bookmark2) 2
  3. [Инди идуальное задание](#_bookmark3) 2
  4. [Характеристики системь](#_bookmark4) 2
  5. [Системнье ьзо ь](#_bookmark5) 2
     1. [Перех ат системньх ьзо о](#_bookmark6) 3
     2. [Системньй ьзо sys\_getpid](#_bookmark7) 4
     3. [Системньй ьзо sys\_setpid](#_bookmark8) 7
     4. [Системньй ьзо sys\_chdir](#_bookmark9) 7
     5. [Системньй ьзо sys\_sysinfo](#_bookmark10) 10
  6. [Вь од](#_bookmark11) 14

Лабораторная работа №1

## Цель работы

Ознакомиться с системньми ьзо ами, научиться разрабать ать нерех атчики для системньх ьзь о и носить изменения ядро.

## Программа работы

Для каждо,о системно,о ьзо а из инди идуально,о задания:

* + - При ести онисание.
    - Разработать нро,рамму с нримером иснользо ания.
    - Осущест ить нерех ат ьзо а черех хук.
    - Проанализиро ать исходньй код ьзо а.

## Индивидуальное задание

Системнье ьзо ь:

* + - sys\_getpid
    - sys\_setpid
    - sys\_chdir
    - sys\_sysinfo

## Характеристики системы

1

201s−05−07 0 2 : 5 7 ( : 0 )

n i k i t a © n i k i t a −V i r t u a l B o x : - $ c a t / p r o c / v e r s i o n

L i n u x v e r s i o n 4.4.0 −121 − g e n e r i c ( b u i l d d © l c y 0 1 −amd64 −004) ( g c c v e r s i o n 5 . 4 . 0 20 160 609 ( Ubuntu 5 . 4 . 0 − 6 u b u n tu 1 - 1 6 . 0 4 . 9 ) ) #145−Ubuntu SMP F r i Apr 13 1 3 : 4 7 : 2 3 UTC 201s

n i k i t a t t y 7

n i k i t a © n i k i t a −V i r t u a l B o x : - $ who

2

3

4

## Системные вызовы

В конечном ито,е, ,ла ная задача онерационной системь - это обслужи ание нотребностей нрикладньх нользо ательских нроцессо . И обеснечи ается это обслужи ание через механизм системньх ьзо о .

В любой ( том числе и микроядерной) онерационной системе системньй ьзо ьнолняется некоторой

ьделенной нроцессорной инструкцией, нрерь ающей носледо ательное ьнолнение команд и нередающий унра ление коду режима сунер изора. Это обьчно некая команда нро,раммно,о нрерь ания, за исимости от архитектурь нроцессора разнье ремена это бьли командь с мнемониками ида: svc, ernt, trap, int и им нодобньми [[1](#_bookmark12)]

Командь нрерь ания для ОС, ностроенньх на архитектуре Intel x86:

|  |  |
| --- | --- |
| OS | desc |
| MS-DOS | 21h |
| Windows | 2Eh |
| Linux | 80h |
| QNX | 21h |
| MINIX 3 | 21h |

Прикладной нроцесс ьзь ает требуемье ему службь носредст ом библиотечно,о ьзо а к множест у библиотек либо ида \*.so (динамическое с язь ание), либо нрикомноно ь ая к себе фра,мент из библиотеки

ида \*.a (статическое с язь ание). Самье из естнье нримерь - это стандартная библиотека язька С libc.so или libpthread.so - библиотека POSIX нотоко [[1](#_bookmark12)]

Онисания системньх ьзо о ( отличие от библиотечньх) отнесень к секции 2 руко одст ах rnan. Все системнье ьзо ь далее нреобразуются ьзо ядра функцией syscall(), 1-м нараметром которо,о будет идентификатор ьнолняемо,о системно,о ьзо а, нанример \_\_NR\_execve.

Найдем онисания заданньх системньх ьзо о таблице екторо системньх ьзо о syscall\_32.tbl:

< . . . >

\_\_i a32\_compat\_sy s \_sy si nf o

s y s \_ s y s i n f o

< . . . >

116 i 3 s 6 s y s i n f o

\_\_ia32\_sys\_getpid

s y s \_ g e t p i d

< . . . >

20 i 3 s 6 g e t p i d

\_\_ia32\_sy s\_chdi r

s y s \_ c h d i r

12 i 3 s 6 c h d i r

< . . . >

#

# The f o r m a t i s :

# <number> <a b i > <name> <e n t r y p o i n t > <compat e n t r y p o i n t >

#

# 32− b i t s y s t e m c a l l numbers and e n t r y v e c t o r s

# l i n u x / a r c h / xs6 / e n t r y / s y s c a l l s / s y s c a l l \_ 3 2 . t b l

#

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

### Перехват системных вызовов

Для нерех ата системньх ьзо о оснользуемся библиотекой krnod\_hooking [[2](#_bookmark13)] Данная библиотека со- держит набор системньх функций для ре,истрации и снятия нерех атчико с заданньх функций. Принцин работь осно ь ается на нодмене функции нутри ядра. Синтаксис нерех атчика иллюстрируется следую- щим нримером нерех ата системной функции inode\_perrnission:

#i n c l u d e < l i n u x / f s . h>

< . . . >

DECLARE\_KHOOK( i n o d e \_ p e r m i s s i o n ) ;

{

i n t k h o o k \_ i n o d e \_ p e r m i s s i o n ( s t r u c t i n o d e ∗ i n o d e , i n t mode )

i n t r e s u l t ;

KHOOK\_USAGE\_INC( i n o d e \_ p e r m i s s i o n ) ;

debug ( 11%s (%pK,%0s x ) [% s ] \ n11 , \_\_func\_\_ , i n o d e , mode , c u r r e n t −>comm) ; r e s u l t KHOOK\_ORIGIN( i n o d e \_ p e r m i s s i o n , i n o d e , mode ) ;

debug ( 11%s (%pK,%0s x ) [% s ] %d \n11 , \_\_func\_\_ , i n o d e , mode , c u r r e n t −>comm , r e s u l t ) ;

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

|  |  |  |
| --- | --- | --- |
| 17 |  |  |
| 18 |  | KHOOK\_USAGE\_DEC( i n o d e \_ p e r m i s s i o n ) ; |
| 19 |  |  |
| 20 |  | r e t u r n r e s u l t ; |
| 21 | } |  |
| 22 |  |  |
| 23 | < . . . > | |

DECLARE\_КНOOК указь ает на то, какие функции я ляются нерех атчиками. КНOOК\_USAGE\_INC и КНOOК\_USAGE\_DEC онределяют область, которой ьнолняется хук. КНOOК\_ORIGIN ьзь ает неносредст енно системньй ьзо с заданньми ар,ументами.

Доба ление собст енно,о кода ядро нроиз одится с номощью модулей. Модули раснола,аются дирек- тории /lib/rnodules/$(unarne -r). Сборка библиотеки нроиз одится нри номощи следующе,о rnakefile:

NAME

: h o o k i n g

o b j −m : $ (NAME) . o

o b j −y : l i b u d i s s 6 /

l d f l a g s −y : −T$ ( s r c ) / l a y o u t . l d s

$ (NAME)−y : module− i n i t . o l i b u d i s s 6 / b u i l t −i n . o a l l :

make −C / l i b / m o d u l e s /$ ( s h e l l uname −r ) / b u i l d M $ ( s h e l l pwd )

c l e a n :

make −C / l i b / m o d u l e s /$ ( s h e l l uname −r ) / b u i l d M $ ( s h e l l pwd ) c l e a n

1

2

3

4

5

6

7

8

9

10

11

12

13

14

Доба ление но о,о модуля систему нроиз одится через утилиту insrnod:

n i k i t a © n i k i t a −V i r t u a l B o x : -/ f i l e s / s t u d e n t / temp / kmod\_hooking$ s u d o i n s m o d h o o k i n g . ko

1

Удаление модуля из систему нроиз одится через утилиту rrnrnod:

n i k i t a © n i k i t a −V i r t u a l B o x : -/ f i l e s / s t u d e n t / temp / kmod\_hooking$ s u d o rmmod h o o k i n g

1

### Системный вызов sys getpid

Системньй ьзо sys\_getpid() оз ращает идентификатор текуще,о нроцесса. Пользо ательская функ- ция getpid() нредоста ляет достун к данному системному ьзо у и объя ляется за,оло очном файле unistd.h.

Пример использования

Функция не нринимает никаких ар,ументо и оз ращает целочисленньй идентификатор нроцесса:

#i n c l u d e <i o s t r e a m >

#i n c l u d e <u n i s t d . h>

i n t main ( ) {

s t d : : c o u t << 11 P r o c e s s PID : 11 << g e t p i d ( ) << s t d : : e n d l ; r e t u r n 0 x0 ;

}

1

2

3

4

5

6

7

Результат ьнолнения нро,раммь:

. / e x e c u t a b l e

[ 1 ] + Done

n i k i t a © n i k i t a −V i r t u a l B o x : -/ f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s $ . / e x e c u t a b l e &

[ 1 ] 9225

P r o c e s s PID : 9225

1

2

3

4

Отслеживание системного вызова

Понробуем отследить системньй ьзо нредьдущей нро,рамме через утилиту strace:

) 1s

e x i t \_ g r o u p ( 0 ) 7

+++ e x i t e d w i t h 0 +++

0 x 1 4 1 s0 0 0

0 x 1 4 4 a0 0 0

9225

s t \_ r d e v makedev ( 1 3 6 , 1 9 ) , . . . } ) 0 1s P r o c e s s PID : 9225

f s t a t ( 1 , { st\_mode S\_IFCHR I 0 6 2 0 ,

w r i t e ( 1 , 11 P r o c e s s PID : 9225\ n11 ,

n i k i t a © n i k i t a −V i r t u a l B o x : -/ f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s $ s t r a c e . / e x e c u t a b l e

e x e c v e ( 11 . / e x e c u t a b l e 11 , [ 11 . / e x e c u t a b l e 11 ] , [ \_∗ 74 V а r 5 ∗\_ ] ) 0

< . . . >

b r k (NULL ) b r k ( 0 x 1 4 4 a0 0 0 ) g e t p i d ( )

1

2

3

4

5

6

7

8

9

10

11

12

13

После создания нроцесса ьзь ается функция getpid и ь одится на экран функцией write, носле че,о нро,рамма за ершается.

Перехват системного вызова

Реализуем нерех ат системно,о ьзо а sys\_getpid с номощью krnod\_hooking:

#i n c l u d e < l i n u x / s y s c a l l s . h>

< . . . >

DECLARE\_KHOOK( s y s \_ g e t p i d ) ; l o n g k h o o k \_ s y s \_ g e tp i d ( )

{

KHOOK\_USAGE\_INC( s y s \_ g e t p i d ) ;

l o n g r e s u l t KHOOK\_ORIGIN( s y s \_ g e t p i d ) ;

p r i n t k ( 11 S y s c a l l ' s y s \_ g e t p i d ' ho o k e d . R e s u l t : %d . \ n11 , r e s u l t ) ; KHOOK\_USAGE\_DEC( s y s \_ g e t p i d ) ;

r e t u r n r e s u l t ;

}

< . . . >

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

Данньй код доба ляется linux как модуль ядра, а результат работь журналируется файл kern.log.

Убедимся нра ильности работь хука:

S y s c a l l ' s y s \_ g e t p i d ' ho o k e d .

0 s : 2 0 : 3 3 n i k i t a −V i r t u a l B o x

R e s u l t : 9 4 3 1 .

a v x : 2 0 0 1 0 . 0 0 0 MB/ s e c

B t r f s l o a d e d

[ 1 1 6 0 9 . 7 3 2 7 1 9 ]

[ 1 1 6 0 9 . 7 4 9 9 3 0 ] [ 1 9 3 s 6 . s 3 4 5 4 3 ]

May 3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x k e r n e l :

May 3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x k e r n e l : May 3 k e r n e l :

k e r n e l : [ 1 1 6 0 9 . 6 9 3 s 0 2 ] x o r : a u t o m a t i c a l l y u s i n g b e s t

3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x

checksumming f u n c t i o n :

May

a l g o r i t h m

k e r n e l : [ 1 1 6 0 9 . 6 9 2 7 0 4 ] r a i d 6 : u s i n g s s s e 3 x 2 r e c o v e r y

3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x

May

rmw e n a b l e d

. . . . x o r ( ) 105s9 MB/ s ,

r a i d 6 :

[ 1 1 6 0 9 . 6 9 2 7 0 3 ]

k e r n e l :

3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x

u s i n g a l g o r i t h m s s e 2 x 4

May

0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x

gen ( ) 153s9 MB/ s

s s e 2 x 2 x o r ( ) s6ss MB/ s

s s e 2 x 4 gen ( ) 153s9 MB/ s s s e 2 x 4 x o r ( ) 105s9 MB/ s

r a i d 6 :

r a i d 6 : r a i d 6 : r a i d 6 :

k e r n e l : [ 1 1 6 0 9 . 5 5 7 0 4 6 ]

0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x k e r n e l : [ 1 1 6 0 9 . 6 2 5 1 0 1 ]

0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x k e r n e l : [ 1 1 6 0 9 . 6 9 2 7 0 0 ]

k e r n e l : [ 1 1 6 0 9 . 6 9 2 7 0 2 ]

3

3

3

3 0 6 : 1 0 : 5 5 n i k i t a −V i r t u a l B o x

t a i l / v a r / l o g / k e r n . l o g

n i k i t a © n i k i t a −V i r t u a l B o x : -/ f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s $

May

May May May

. / e x e c u t a b l e

Done

. / e x e c u t a b l e &

[ 1 ] +

n i k i t a © n i k i t a −V i r t u a l B o x : -/ f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s $

[ 1 ] 9431

P r o c e s s PID : 9431

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

После зануска нро,раммь kern.log ноя илась занись о ноньтке достуна к системному ьзо у sys\_getpid.

Анализ исходного кода

Прототин системно,о ьзо а sys\_getpid находится include/linux/syscalls.h исходньх файлах ядра linux

[[3](#_bookmark14)]:

\_\_ i n с l u d e \_ l i n u x \_ 5 y 5 с а l l 5 . h

a s m l i n k a g e l o n g s y s \_ g e t p i d ( v o i d ) ;

1

2

3

Неносредст енно реализация sys\_getpid находится kernel/sys.c исходньх файлах ядра linux [[4](#_bookmark15)]:

\_\_ k e r n e l \_ 5 y 5 . с

SYSCALL\_DEFINE0 ( g e t p i d )

{

r e t u r n t a s k \_ t g i d \_ v n r ( c u r r e n t ) ;

}

1

2

3

4

5

6

Можно заметить, что ьзь атся функция ядра task\_tgid\_vnr, которая с ою очередь ьзь ает \_\_task\_pid\_nr

kernel/pid.c исходньх файлах ядра linux [[5](#_bookmark16)]:

1

\_\_ k e r n e l \_ p i d . с

p i d \_t \_\_task\_pid\_nr\_ns ( s t r u c t t a s k \_ s t r u c t ∗ t a s k , enum p i d \_ t y p e ty p e ,

{

s t r u c t pi d\_names pac e ∗ n s )

p i d \_t n r 0 ;

r c u \_ r e a d \_ l o c k ( ) ; i f ( ! n s )

n s t a s k \_ a c t i v e \_ p i d \_ n s ( c u r r e n t ) ; i f ( l i k e l y ( p i d \_ a l i v e ( t a s k ) ) ) {

i f ( t y p e ! PIDTYPE\_PID ) {

i f ( t y p e \_\_PIDTYPE\_TGID) t y p e PIDTYPE\_PID ;

}

t a s k t a s k −>g r o u p \_ l e a d e r ;

}

n r pid\_nr\_ns ( r c u \_ d e r e f e r e n c e ( t a s k −>p i d s [ t y p e ] . p i d ) , n s ) ;

r c u \_ r e a d \_ u n l o c k ( ) ; r e t u r n n r ;

}

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

На ремя работь функции блокируется чтение, нро еряется наличие текуще,о нроцесса и наличие gpid, носле че,о ьзь ается функция pid\_nr\_ns:

}

n r u p i d −>n r ;

r e t u r n n r ;

}

i f ( p i d && ns−> l e v e l < p i d −> l e v e l ) {

u p i d &p i d −>numbers [ ns−> l e v e l ] ; i f ( u p i d −>n s n s )

s t r u c t u p i d ∗ u p i d ;

p i d \_t n r 0 ;

p i d \_t pid\_nr\_ns ( s t r u c t p i d ∗ p i d , s t r u c t pi d\_names pac e ∗ n s )

{

\_\_ k e r n e l \_ p i d . с

1

2

3

4

5

6

7

8

9

10

11

12

13

14

Если оба ар,умента (структурь pid и pid\_narnespace) корректнь, то оз ращается pid текуще,о нроцесса,

нроти ном случае оз ращается 0.

### Системный вызов sys setpid

Тако,о системно,о ьзо а, оче идно, не сущест ует, нотому что идентификатор нроцесса задается еди- нождь нри зануске нроцесса.

Для ьделения но о,о pid иснользуется функция alloc\_pid:

1

\_\_ k e r n e l \_ p i d . с

2

s t r u c t p i d ∗ a l l o c \_ p i d ( s t r u c t pi d\_names pac e ∗ n s )

< . . . >

3

4

{

5

6

p i d kmem\_cache\_alloc ( ns−>p i d \_c ac he p , GFP\_KERNEL) ;

7

8

i f ( ! p i d )

r e t u r n ERR\_PTR( r e t v a l ) ;

9

10

11

tmp n s ;

p i d −> l e v e l ns−> l e v e l ;

12

13

f o r ( i ns−> l e v e l ; i > 0 ; i −−) {

14

15

i n t pid\_min 1 ;

16

17

i d r \_ p r e l o a d (GFP\_KERNEL) ;

s p i n \_ l o c k \_ i r q (& pi dmap\_l o c k ) ;

18

19

\_∗

∗ i n i t r e а l l y n e e d 5 p i d 1 , b u t а f t e r r e а с h i n g t h e mаximum

∗ wrаp b ас k t o RESERVE0\_Р/0S

∗\_

20

21

22

23

i f ( i d r \_ g e t \_ c u r s o r (&tmp−>i d r ) > RESERVED\_PIDS )

24

25

pid\_min RESERVED\_PIDS ;

\_∗

∗ S t o r e а n u l l p o i n t e r 5 o f i n d \_ p i d \_ n 5 d o e 5 n o t

∗ а p а r t i а l l y i n i t i а l i z e d Р/0 ( 5 e e b e l o w ) .

∗\_

26

27

28

f i n d

29

30

n r i d r \_ a l l o c \_ c y c l i c (&tmp−>i d r , NULL , pid\_min ,

31

32

pid\_max , GFP\_ATOMIC) ;

s p i n \_ u n l o c k \_ i r q (& pi dm ap\_l o c k ) ; i d r \_ p r e l o a d \_ e n d ( ) ;

33

34

35

36

i f ( n r < 0 ) {

r e t v a l n r ; g o to o u t \_ f r e e ;

}

p i d −>numbers [ i ] . n r n r ; p i d −>numbers [ i ] . n s tmp ; tmp tmp−>p a r e n t ;

}

< . . . >

37

38

39

40

41

42

43

44

45

46

47

}

Функция ьзь ает krnern\_cache\_alloc с ар,ументом иде набора кэширо анньх структур для ьделе- ния намяти нод структуру но о,о pid [[6](#_bookmark17)].

Далее нроисходит инициализация нолей структурь за исимости от уро ня ар,умента pid\_narnespace. Функция idr\_alloc\_cyclic ньтается ьделить но ьй идентификатор нроцесса. Если нолучилось, то иденти- фикатор занись ается структуру.

### Системный вызов sys chdir

Системньй ьзо sys\_chdir() задает текущий нуть ьнолнения нроцесса. Пользо ательская функция chdir() нредоста ляет достун к данному системному ьзо у и объя ляется за,оло очном файле unistd.h.

Пример использования

Для нримера ь едем текущий нуть ьнолнения нроцесса, номеняем е,о функцией chdir() и ь едем нуть еще раз. Путь должен измениться на ар,умент функции chdir():

#i n c l u d e <i o s t r e a m >

#i n c l u d e <u n i s t d . h>

s t a t i c c o n s t i n t DIRECTORY\_PATH\_SIZE 1 0 2 4 ; s t a t i c c o n s t c h a r ∗ DIRECTORY\_PATH 11 /home11 ;

s t a t i c c h a r c u r r e n t D i r e c t o r y [ DIRECTORY\_PATH\_SIZE ] ; c o n s t c h a r ∗ g e t C u r r e n t D i r e c t o r y ( ) ;

i n t main ( ) {

s t d : : c o u t << 11 C u r r e n t d i r e c t o r y : 11 << g e t C u r r e n t D i r e c t o r y ( ) << s t d : : e n d l ; c h d i r (DIRECTORY\_PATH) ;

s t d : : c o u t << 11 C u r r e n t d i r e c t o r y a f t e r c h d i r : 11 << g e t C u r r e n t D i r e c t o r y ( ) << s t d : : e n d l ; r e t u r n 0 x0 ;

}

c o n s t c h a r ∗ g e t C u r r e n t D i r e c t o r y ( ) {

c h a r ∗ r e s u l t g e tc w d ( c u r r e n t D i r e c t o r y , DIRECTORY\_PATH\_SIZE) ;

i f ( r e s u l t NULL ) { r e t u r n NULL ;

}

r e t u r n c u r r e n t D i r e c t o r y ;

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

Результат ьнолнения нро,раммь:

n i k i t a © n i k i t a −V i r t u a l B o x : -/ f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s $ . / e x e c u t a b l e

C u r r e n t d i r e c t o r y : /home/ n i k i t a / f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s

C u r r e n t d i r e c t o r y a f t e r c h d i r : /home

1

2

3

Отслеживание системного вызова

Понробуем отследить системньй ьзо нредьдущей нро,рамме через утилиту strace:

n i k i t a © n i k i t a −V i r t u a l B o x : -/ f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s $ s t r a c e . / e x e c u t a b l e

e x e c v e ( 11 . / e x e c u t a b l e 11 , [ 11 . / e x e c u t a b l e 11 ] , [ \_∗ 74 V а r 5 ∗\_ ] ) 0

< . . . >

g e tc w d ( 11 /home/ n i k i t a / f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s 11 , 1 0 2 4 ) 49 f s t a t ( 1 , { st\_mode S\_IFCHR I 0 6 2 0 , s t \_ r d e v makedev ( 1 3 6 , 1 9 ) , . . . } ) 0

w r i t e ( 1 , 11 C u r r e n t d i r e c t o r y : /home/ n i k i t a / 11 . . . , 6s C u r r e n t d i r e c t o r y : /home/ n i k i t a / f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s

) 6s

. . . , 37 C u r r e n t d i r e c t o r y a f t e r c h d i r : /home

1

2

3

4

5

6

7

8

9

|  |  |  |
| --- | --- | --- |
| 10 | c h d i r ( 11 /home11 ) | 0 |
| 11 | g e tc w d ( 11 /home11 , 1 0 2 4 ) | 6 |
| 12 | w r i t e ( 1 , 11 C u r r e n t d i r e c t o r y a f | t e r c h d i r : / 11 |
| 13 | ) 37 |  |
| 14 | e x i t \_ g r o u p ( 0 ) | 7 |
| 15 | +++ e x i t e d w i t h 0 +++ | |

Как и ожидалось, бьла ьз ана ценочка системньх ьзо о : getcwd -> write -> chdir -> getcwd -> write,

результате которой изменилось значение нуть ьнолнения нроцесса.

Перехват системного вызова

Реализуем нерех ат системно,о ьзо а sys\_chdir с номощью krnod\_hooking:

1

#i n c l u d e < l i n u x / s y s c a l l s . h>

< . . . >

DECLARE\_KHOOK( s y s \_ c h d i r ) ;

{

l o n g k h o o k \_ s y s \_ c h d i r ( c o n s t c h a r \_\_user ∗ f i l e n a m e )

KHOOK\_USAGE\_INC( s y s \_ c h d i r ) ;

l o n g r e s u l t KHOOK\_ORIGIN( s y s \_ c h d i r , f i l e n a m e ) ;

p r i n t k ( 11 S y s c a l l ' s y s \_ c h d i r ' ho o k e d . A rguments : %s . R e s u l t : %d . \ n 11 , f i l e n a m e , r e s u l t ) ; KHOOK\_USAGE\_DEC( s y s \_ c h d i r ) ;

r e t u r n r e s u l t ;

}

< . . . >

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

Данньй код доба ляется linux как модуль ядра, а результат работь журналируется файл kern.log.

Убедимся нра ильности работь хука:

A rguments : /home . R e s u l t : 0 .

k e r n e l : [ 1 9 4 4 5 . 9 2 3 4 3 2 ] S y s c a l l ' s y s \_ c h d i r ' ho o k e d .

3 0 s : 2 1 : 3 2 n i k i t a −V i r t u a l B o x

May

R e s u l t : 9 4 3 1 .

S y s c a l l ' s y s \_ g e t p i d ' ho o k e d .

0 s : 2 0 : 3 3 n i k i t a −V i r t u a l B o x

a v x : 2 0 0 1 0 . 0 0 0 MB/ s e c

B t r f s l o a d e d

[ 1 1 6 0 9 . 7 3 2 7 1 9 ]

[ 1 1 6 0 9 . 7 4 9 9 3 0 ] [ 1 9 3 s 6 . s 3 4 5 4 3 ]

May 3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x k e r n e l :

May 3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x k e r n e l : May 3 k e r n e l :

checksumming f u n c t i o n :

k e r n e l : [ 1 1 6 0 9 . 6 9 3 s 0 2 ] x o r : a u t o m a t i c a l l y u s i n g b e s t

3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x

May

a l g o r i t h m

k e r n e l : [ 1 1 6 0 9 . 6 9 2 7 0 4 ] r a i d 6 : u s i n g s s s e 3 x 2 r e c o v e r y

3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x

May

rmw e n a b l e d

k e r n e l : [ 1 1 6 0 9 . 6 9 2 7 0 3 ] r a i d 6 : . . . . x o r ( ) 105s9 MB/ s ,

3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x

May

gen ( ) 153s9 MB/ s

u s i n g a l g o r i t h m s s e 2 x 4

3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x

gen ( ) 153s9 MB/ s

x o r ( ) 105s9 MB/ s

s s e 2 x 4

s s e 2 x 4

r a i d 6 :

r a i d 6 : r a i d 6 :

[ 1 1 6 0 9 . 6 2 5 1 0 1 ]

[ 1 1 6 0 9 . 6 9 2 7 0 0 ]

[ 1 1 6 0 9 . 6 9 2 7 0 2 ]

k e r n e l :

k e r n e l : k e r n e l :

t a i l / v a r / l o g / k e r n . l o g

3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x

3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x

n i k i t a © n i k i t a −V i r t u a l B o x : -/ f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s $

May

May May

n i k i t a © n i k i t a −V i r t u a l B o x : -/ f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s $ . / e x e c u t a b l e

C u r r e n t d i r e c t o r y : /home/ n i k i t a / f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s

C u r r e n t d i r e c t o r y a f t e r c h d i r : /home

1

2

3

4

5

6

7

8

9

10

11

12

13

14

После зануска нро,раммь kern.log ноя илась занись о ноньтке достуна к системному ьзо у sys\_chdir.

Анализ исходного кода

Прототин системно,о ьзо а sys\_chdir находится include/linux/syscalls.h исходньх файлах ядра linux

[[3](#_bookmark14)]:

\_\_ i n с l u d e \_ l i n u x \_ 5 y 5 с а l l 5 . h

a s m l i n k a g e l o n g s y s \_ c h d i r ( c o n s t c h a r \_\_user ∗ f i l e n a m e ) ;

1

2

3

Неносредст енно реализация sys\_chdir находится kernel/sys.c исходньх файлах ядра linux [[4](#_bookmark15)]:

}

r e t u r n k s y s \_ c h d i r ( f i l e n a m e ) ;

{

SYSCALL\_DEFINE1 ( c h d i r , c o n s t c h a r \_\_user ∗ , f i l e n a m e )

\_\_ k e r n e l \_ 5 y 5 . с

1

2

3

4

5

6

Можно заметить, что ьзь атся функция ядра ksys\_chdir fs/open.c исходньх файлах ядра linux [[7](#_bookmark18)]:

\_\_ f 5 \_ open . с

1

2

3

{

i n t k s y s \_ c h d i r ( c o n s t c h a r \_\_user ∗ f i l e n a m e )

s t r u c t p a t h p a t h ; i n t e r r o r ;

u n s i g n e d i n t l o o k u p \_ f l a g s LOOKUP\_FOLLOW I LOOKUP\_DIRECTORY ; r e t r y :

e r r o r u s e r \_p ath \_at (AT\_FDCWD, f i l e n a m e , l o o k u p \_ f l a g s , &p a t h ) ; i f ( e r r o r )

g o to o u t ;

e r r o r i n o d e \_ p e r m i s s i o n ( p a t h . d e n t r y −>d\_inode , MAY\_EХEC I MAY\_CHDIR) ;

i f ( e r r o r )

g o to dput\_and\_out ;

set\_fs\_pwd ( c u r r e n t −>f s , &p a t h ) ;

dput\_and\_out : path\_put(& p a t h ) ;

i f ( r e t r y \_ e s t a l e ( e r r o r , l o o k u p \_ f l a g s ) ) { l o o k u p \_ f l a g s I LOOKUP\_REVAL ;

g o to r e t r y ;

}

o u t :

r e t u r n e r r o r ;

}

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

Функция нро еряет наличие зада аемо,о нути и наличие нра на е,о изменение. После это,о ьзь ается функция set\_fs\_pwd, которая неносредст енно задает нуть:

}

v o i d set\_fs\_pwd ( s t r u c t f s \_ s t r u c t ∗ f s , c o n s t s t r u c t p a t h ∗ p a t h )

s t r u c t p a t h old\_pwd ;

p ath \_g e t ( p a t h ) ; s p i n \_ l o c k (& f s −>l o c k ) ; w r i t e \_ s e q c o u n t \_ b e g i n (& f s −>s e q ) ; old\_pwd f s −>pwd ;

f s −>pwd ∗ p a t h ; w r i t e \_ s e q c o u n t \_ e n d (& f s −>s e q ) ; s p i n \_ u n l o c k (& f s −>l o c k ) ;

i f ( old\_pwd . d e n t r y ) path\_put(&old\_pwd ) ;

{

\_\_ f 5 \_ open . с

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

### Системный вызов sys sysinfo

Системньй ьзо sys\_sysinfo() ь одит системную информацию, такую как: ремя с момента зануска системь секундах, количест о намяти и количест о с ободно намяти байтах, количест о акти ньх нро- цессо , размер с фна и др. Пользо ательская функция sysinfo() нредоста ляет достун к данному системному

ьзо у и объя ляется за,оло очном файле sys/sysinfo.h.

Пример использования

Для нримера ь едем системную информацию удобном для чтения формате:

#i n c l u d e <i o s t r e a m >

#i n c l u d e <s y s / s y s i n f o . h>

s t a t i c

s t a t i c s t a t i c s t a t i c

c o n s t i n t MEGABYTE\_IN\_BYTES 1024 ∗ 1 0 2 4 ;

c o n s t i n t MINUTE\_IN\_SECONDS 6 0 ;

c o n s t i n t HOUR\_IN\_SECONDS MINUTE\_IN\_SECONDS ∗ 6 0 ; c o n s t i n t DAY\_IN\_SECONDS HOUR\_IN\_SECONDS ∗ 2 4 ;

1

2

3

4

5

6

7

8

9

i n t main ( ) {

s t r u c t s y s i n f o i n f o r m a t i o n ; s y s i n f o (& i n f o r m a t i o n ) ;

s t d : : c o u t << 11 Time s i n c e b o o t : d a y s 11 << i n f o r m a t i o n . u p ti m e / DAY\_IN\_SECONDS << 11 , 11

<< ( i n f o r m a t i o n . u p ti m e % DAY\_IN\_SECONDS) /

HOUR\_IN\_SECONDS << 11 : 11

<< ( i n f o r m a t i o n . u p ti m e % HOUR\_IN\_SECONDS) /

MINUTE\_IN\_SECONDS << 11 : 11

<< i n f o r m a t i o n . u p ti m e % MINUTE\_IN\_SECONDS <<

s t d : : e n d l ;

s t d : : c o u t << 11 T o t a l RAM: 11 << i n f o r m a t i o n . t o t a l r a m / MEGABYTE\_IN\_BYTES << 11 Mb11 << s t d : : e n d l ;

s t d : : c o u t << 11 F r e e RAM: 11 << i n f o r m a t i o n . f r e e r a m / MEGABYTE\_IN\_BYTES << 11 Mb11 << s t d

: : e n d l ;

s t d : : c o u t << 11 P r o c e s s c o u n t : 11 << i n f o r m a t i o n . p r o c s << s t d : : e n d l ; r e t u r n 0 x0 ;

}

10

11

12

13

14

15

16

17

18

19

20

21

22

Результат ьнолнения нро,раммь:

n i k i t a © n i k i t a −V i r t u a l B o x : -/ f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s $ . / e x e c u t a b l e

Time s i n c e b o o t : d a y s 0 , 1 4 : 4 2 : 2 3

T o t a l RAM: 3951 Mb F r e e RAM: 605 Mb P r o c e s s c o u n t : 631

1

2

3

4

5

Отслеживание системного вызова

Понробуем отследить системньй ьзо нредьдущей нро,рамме через утилиту strace:

w r i t e ( 1 , 11 F r e e RAM: 599 Mb\n11 , 17 F r e e RAM: 599 Mb

) 17

w r i t e ( 1 , 11 P r o c e s s c o u n t : 633\ n 11 , 19 P r o c e s s c o u n t : 633

) 19

e x i t \_ g r o u p ( 0 ) 7

+++ e x i t e d w i t h 0 +++

19

11 T o t a l RAM: 3951 Mb\n11 , 19 T o t a l RAM: 3951 Mb

)

{ st\_mode S\_IFCHR I 0 6 2 0 , s t \_ r d e v makedev ( 1 3 6 , 1 9 ) , . . . } ) 0

11 Time s i n c e b o o t : d a y s 0 , 1 4 : 4 3 : 5 11 . . . , 34 Time s i n c e b o o t : d a y s 0 , 1 4 : 4 3 : 5 0

f s t a t ( 1 ,

w r i t e ( 1 ,

) 34

w r i t e ( 1 ,

n i k i t a © n i k i t a −V i r t u a l B o x : -/ f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s $ s t r a c e . / e x e c u t a b l e

e x e c v e ( 11 . / e x e c u t a b l e 11 , [ 11 . / e x e c u t a b l e 11 ] , [ \_∗ 74 V а r 5 ∗\_ ] ) 0

< . . . >

s y s i n f o ({ u p ti m e 53030 , l o a d s [ 3 1 3 2 s , 2 9 s2 4 , 2 0 1 6 0 ] , t o t a l r a m 4142927s72 , f r e e r a m

62s19532s , s h a r e d r a m 6796492s , b u f f e r r a m 390s3212s , t o t a l s w a p 4292s6604s , f r e e s w a p

4292s6604s , p r o c s 633 , t o t a l h i g h 0 , f r e e h i g h 0 , mem\_unit 1}) 0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

Как и ожидалось бьла нолучена структура sysinfo через соот етст ующий системньй ьзо .

Перехват системного вызова

Реализуем нерех ат системно,о ьзо а sys\_sysinfo с номощью krnod\_hooking:

#i n c l u d e < l i n u x / s y s c a l l s . h>

< . . . >

DECLARE\_KHOOK( s y s \_ s y s i n f o ) ;

{

l o n g k h o o k \_ s y s \_ s y s i n f o ( s t r u c t s y s i n f o \_\_user ∗ i n f o )

KHOOK\_USAGE\_INC( s y s \_ s y s i n f o ) ;

1

2

3

4

5

6

7

8

9

10

l o n g r e s u l t KHOOK\_ORIGIN( s y s \_ s y s i n f o , i n f o ) ;

p r i n t k ( 11 S y s c a l l ' s y s \_ s y s i n f o ' ho o k e d . A rguments : %l d , %l d , %l d , %d . R e s u l t : %d . \ n11 , i n f o −>upti m e , i n f o −>t o t a l r a m , i n f o −>f r e e r a m , i n f o −>p r o c s , r e s u l t ) ;

KHOOK\_USAGE\_DEC( s y s \_ s y s i n f o ) ; r e t u r n r e s u l t ;

}

< . . . >

11

12

13

14

15

16

17

18

19

Данньй код доба ляется linux как модуль ядра, а результат работь журналируется файл kern.log.

Убедимся нра ильности работь хука:

. / e x e c u t a b l e

Time s i n c e b o o t : d a y s 0 , 5 : 2 4 : 3 3

t a i l / v a r / l o g / k e r n . l o g

3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x

s s e 2 x 4 x o r ( ) 105s9 MB/ s

u s i n g a l g o r i t h m s s e 2 x 4

. . . . x o r ( ) 105s9 MB/ s ,

mw e n a b l e d

u s i n g s s s e 3 x 2 r e c o v e r y

a l g o r i t h m

May 3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x

k e r n e l :

[ 1 1 6 0 9 . 6 9 3 s 0 2 ] x o r : a u t o m a t i c a l l y u s i n g b e s t

May

May May

checksumming f u n c t i o n :

3

3

3

0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x

0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x

k e r n e l :

k e r n e l : k e r n e l :

[ 1 1 6 0 9 . 7 3 2 7 1 9 ]

[ 1 1 6 0 9 . 7 4 9 9 3 0 ] [ 1 9 3 s 6 . s 3 4 5 4 3 ]

a v x : 2 0 0 1 0 . 0 0 0 MB/ s e c

B t r f s l o a d e d

0 s : 2 0 : 3 3 n i k i t a −V i r t u a l B o x

S y s c a l l ' s y s \_ g e t p i d ' ho o k e d .

R e s u l t : 9 4 3 1 .

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1  2  3  4  5  6 | n i k i t a © n i k i t a −V i r t u a l B o x : -/ f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s $  T o t a l RAM: 3951 Mb F r e e RAM: 20s Mb P r o c e s s c o u n t : 644  n i k i t a © n i k i t a −V i r t u a l B o x : -/ f i l e s / s t u d e n t / ezsem4 / s y s /1/ l i s t i n g s $ | | | | |
| 7 | May |  | k e r n e l : | [ 1 1 6 0 9 . 6 9 2 7 0 0 ] | r a i d 6 : |
| 8 | May | 3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x  gen ( ) 153s9 MB/ s | k e r n e l : | [ 1 1 6 0 9 . 6 9 2 7 0 2 ] | r a i d 6 : |
| 9 | May  r | 3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x | k e r n e l : | [ 1 1 6 0 9 . 6 9 2 7 0 3 ] | r a i d 6 : |
| 10 | May | 3 0 6 : 1 0 : 5 6 n i k i t a −V i r t u a l B o x | k e r n e l : | [ 1 1 6 0 9 . 6 9 2 7 0 4 ] | r a i d 6 : |

11

12

13

14

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 15 | May | 3 0 s : 2 1 : 3 2 n i k i t a −V i r t u a l B o x  A rguments : /home . R e s u l t : 0 . | k e r n e l : | [ 1 9 4 4 5 . 9 2 3 4 3 2 ] | S y s c a l l | ' s y s \_ c h d i r ' ho o k e d . |
| 16 | May | 3 0 s : 2 2 : 0 1 n i k i t a −V i r t u a l B o x A rguments : 1 9 4 7 3 , 4 1 4 2 9 2 3 7 7 6 , | k e r n e l : 2 1 s1 0 3 s0 | [ 1 9 4 7 3 . 3 2 4 4 2 9 ]  s , 6 4 4 . R e s u l t : | S y s c a l l 0 . | ' s y s \_ s y s i n f o ' ho o k e d . |

После зануска нро,раммь kern.log ноя илась занись о ноньтке достуна к системному ьзо у sys\_sysinfo.

Анализ исходного кода

Прототин системно,о ьзо а sys\_sysinfo находится include/linux/syscalls.h исходньх файлах ядра linux

[[3](#_bookmark14)]:

\_\_ i n с l u d e \_ l i n u x \_ 5 y 5 с а l l 5 . h

a s m l i n k a g e l o n g s y s \_ s y s i n f o ( s t r u c t s y s i n f o \_\_user ∗ i n f o ) ;

1

2

3

Неносредст енно реализация sys\_sysinfo находится kernel/sys.c исходньх файлах ядра linux [[4](#_bookmark15)]:

}

s t r u c t s y s i n f o v a l ;

d o \_ s y s i n f o (& v a l ) ;

i f ( c o py \_to \_us e r ( i n f o , &v a l , s i z e o f ( s t r u c t s y s i n f o ) ) ) r e t u r n −EFAULT ;

r e t u r n 0 ;

{

SYSCALL\_DEFINE1 ( s y s i n f o , s t r u c t s y s i n f o \_\_user ∗ , i n f o )

\_\_ k e r n e l \_ 5 y 5 . с

1

2

3

4

5

6

7

8

9

10

11

12

13

Можно заметить, что ьзь ается функция ядра do\_sysinfo kernel/sys.c исходньх файлах ядра linux

[[4](#_bookmark15)]:

1 \_\_ k e r n e l \_ 5 y 5 . с

2

3 s t a t i c i n t d o \_ s y s i n f o ( s t r u c t s y s i n f o ∗ i n f o )

4 {

5 u n s i g n e d l o n g mem\_total , s a v \_ t o t a l ;

6 u n s i g n e d i n t mem\_unit , b i t c o u n t ;

7 s t r u c t t i m e s p e c t p ;

8

9 memset ( i n f o , 0 , s i z e o f ( s t r u c t s y s i n f o ) ) ;

10

11 g e t\_m o n o to n i c \_b o o tti m e (& t p ) ;

12 i n f o −>u p ti m e t p . tv \_ s e c + ( t p . tv \_ n s e c 7 1 : 0 ) ;

13

14 g e t\_ a v e n r u n ( i n f o −>l o a d s , 0 , SI\_LOAD\_SHIFT − FSHIFT ) ;

15

16 i n f o −>p r o c s n r \_ t h r e a d s ;

17

18 s i \_m e m i nf o ( i n f o ) ;

19 s i \_ s w a p i n f o ( i n f o ) ;

20

21 \_∗

22 ∗ / f t h e 5um o f а l l t h e а V а i l а b l e memory ( i . e . rаm + 5wаp )

23 ∗ i 5 l e 5 5 t h а n с аn be 5 t o r e d i n а 32 b i t u n 5 i g n e d l o n g t h e n

24 ∗ we с аn be b i n а r y с o m p а t i b l e w i t h 2 . 2 . x k e r n e l 5 . / f not ,

25 ∗ w e l l , i n t h а t с а 5 e 2 . 2 . x wа5 b r o k e n аny w аy 5 . . .

26 ∗

27 ∗ − E r i k A n d e r 5 e n <а n d e r 5 e e @ d e b i а n . o r g >

28 ∗\_

29

30 mem\_total i n f o −>t o t a l r a m + i n f o −>t o t a l s w a p ;

31 i f ( mem\_total < i n f o −>t o t a l r a m I I mem\_total < i n f o −>t o t a l s w a p )

32 g o to o u t ;

33 b i t c o u n t 0 ;

34 mem\_unit i n f o −>mem\_unit ;

35 w h i l e ( mem\_unit > 1 ) {

36 b i t c o u n t ++;

37 mem\_unit >> 1 ;

38 s a v \_ t o t a l mem\_total ;

39 mem\_total << 1 ;

40 i f ( mem\_total < s a v \_ t o t a l )

41 g o to o u t ;

42 }

43

44 \_∗

45 ∗ / f mem\_totаl d i d n o t o V e r f l o w , m u l t i p l y а l l memory V а l u e 5 by

46 ∗ i n f o −>mem\_unit аnd 5 e t i t t o 1 . T h i 5 l e а V e 5 t h i n g 5 с o m p а t i b l e

47 ∗ w i t h 2 . 2 . x , аnd а l 5 o r e t а i n 5 с o m p а t i b i l i t y w i t h e а r l i e r 2 . 4 . x

48 ∗ k e r n e l 5 . . .

49 ∗\_

50

51 i n f o −>mem\_unit 1 ;

52 i n f o −>t o t a l r a m << b i t c o u n t ;

53 i n f o −>f r e e r a m << b i t c o u n t ;

54 i n f o −>s h a r e d r a m << b i t c o u n t ;

55 i n f o −>b u f f e r r a m << b i t c o u n t ;

56 i n f o −>t o t a l s w a p << b i t c o u n t ;

57 i n f o −>f r e e s w a p << b i t c o u n t ;

58 i n f o −>t o t a l h i g h << b i t c o u n t ;

59 i n f o −>f r e e h i g h << b i t c o u n t ;

60

61 o u t :

62 r e t u r n 0 ;

63 }

В нер ую очередь структура обнуляется, носле че,о собирается информация о ремени, намяти, с оне и количест е нроцессо через соот етст ующие функции get\_rnonolitic\_boottirne, get\_averrun, si\_rnerninfo,

si\_swapinfo. В за ершении делается нонра ка на ерсии ядра 2.2.x с нобито ьм сд и,ом нолей результиру- ющей структурь.

## Вывод

В данной работе бьли изучень системнье ьзо ь, нринцинь нерех ата системньх ьзо о , а также доба ление модулей ядро ОС Linux.

В ОС Linux( ерсия ядра v4.17-rc4) онределено около 384 системньх ьзо о для архитектурь i386. Та- кое количест о системньх ьзо о ноз оляет нро,раммисту нолучить удобньй и нолньй достун ко сем комнонентам ОС рамках нра достуна.

Каждьй системньй ьзо содержит элементь синхронизации там ,де это необходимо для корректной работь усло иях мно,оноточности.

Каждьй разработчик сообщест а может улучшать исходньй код системньх ьзо о и дру,их частей ядра нутем создания pull request GitНub ренозитории ядра Linux [[8](#_bookmark19)]. На се,одняшний день ядро Linux разрослось до такой стенени, что е,о изучение нутем анализа кода малоэффекти но.

Перех ат системньх ьзо о достаточно несложно реализуется нодменой функции нутри ядра, однако, для это,о необходимь нра а сунернользо ателя.

# Литература

1. Часть 5. Системнье ьзо ь [Электронньй ресурс], IBM. - URL: [https://www.ibm.com/developerworks/ ru/library/l-linux\_kernel\_05/index.html](https://www.ibm.com/developerworks/ru/library/l-linux_kernel_05/index.html) (дата обращения: 03.05.2018).
2. Kmod Hooking library,Electronic resource, GitHub. — URL: <https://github.com/milabs/kmod_hooking/>

(online; accessed: 03.05.2018).

1. Linux syscalls.h,Electronic resource, GitHub. — URL: [https://github.com/torvalds/linux/blob/master/ include/linux/syscalls.h](https://github.com/torvalds/linux/blob/master/include/linux/syscalls.h) (online; accessed: 03.05.2018).
2. Linux sys.c,Electronic resource, GitHub. — URL: [https://github.com/torvalds/linux/blob/master/ kernel/sys.c](https://github.com/torvalds/linux/blob/master/kernel/sys.c) (online; accessed: 03.05.2018).
3. Linux pid.c,Electronic resource, GitHub. — URL: [https://github.com/torvalds/linux/blob/master/ kernel/pid.c](https://github.com/torvalds/linux/blob/master/kernel/pid.c) (online; accessed: 03.05.2018).
4. PID Allocation in Linux KernelElectronic resource, Medium Corporation. — URL: [https://medium.com/](https://medium.com/%40gargi_sharma/pid-allocation-in-linux-kernel-dc0c78d14e77)

[@gargi\_sharma/pid-allocation-in-linux-kernel-dc0c78d14e77](https://medium.com/%40gargi_sharma/pid-allocation-in-linux-kernel-dc0c78d14e77) (online; accessed: 03.05.2018).

1. Linux open.c,Electronic resource, GitHub. — URL: [https://github.com/torvalds/linux/blob/master/fs/ open.c](https://github.com/torvalds/linux/blob/master/fs/open.c) (online; accessed: 03.05.2018).
2. Linux,Electronic resource, GitHub. — URL: <https://github.com/torvalds/linux>(online; accessed: 03.05.2018).