به نام خدا

دانشکده مهندسی کامپیوتر

آزمایشگاه سیستم های عامل

آزمایش سوم : برنامه نویسی ماژول های هسته و اشنایی با ساختمان های داده در هسته

اعضای گروه:

حسین تاتار - 40133014

محمد امین فرح بخش - 40131029

اسفند 1403

تمرین اول)

قسمت الف : یک ماژول بنویسید که بتوانید به آن پارامترهایی از نوع short, int, long, string, array به عنوان ورودی بدهید و از آن در کد استفاده کنید (راهنمایی: میتوانید درباره module_param و MODULE_PARM_DES جستجو کنید).

جواب :

برای نوشتن یک ماژول هسته لینوکس که بتواند پارامترهای مختلفی از نوع short, int, long, string و array را دریافت کند، می‌توانیم از ماکروهای module_param و MODULE_PARM_DESC استفاده کنیم. این ماکروها به ما امکان می‌دهند تا پارامترهایی را به ماژول منتقل کنیم و از آن‌ها در کد استفاده کنیم.

*** کد این بخش به با نام MultiTypeModule به فایل پیوست شده است ***

```
hosseintatar@hosseintatar-VMware-Virtual-Platform:~/Desktop/Makefile$ make
make -C /lib/modules/6.11.0-18-generic/build M=/home/hosseintatar/Desktop/Makefile modules
make[1]: Entering directory '/usr/src/linux-headers-6.11.0-18-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: x86_64-linux-gnu-gcc-14 (Ubuntu 14.2.0-4ubuntu2) 14.2.0
  You are using:           gcc-14 (Ubuntu 14.2.0-4ubuntu2) 14.2.0
  CC [M]  /home/hosseintatar/Desktop/Makefile/my_module.o
  MODPOST /home/hosseintatar/Desktop/Makefile/Module.symvers
  CC [M]  /home/hosseintatar/Desktop/Makefile/my_module.mod.o
  LD [M]  /home/hosseintatar/Desktop/Makefile/my_module.ko
  BTF [M] /home/hosseintatar/Desktop/Makefile/my_module.ko
Skipping BTF generation for /home/hosseintatar/Desktop/Makefile/my_module.ko due to unavailability of vmlinux
hosseintatar@hosseintatar-VMware-Virtual-Platform:~/Desktop/Makefile$ sudo insmod my_module.ko my_short=10 my_int=20 my_
long=30 my_string="Hello" my_array=1,2,3,4,5
[sudo] password for hosseintatar:
```

```
hosseintatar@hosseintatar-VMware-Virtual-Platform:~/Desktop/Makefile$ sudo dmesg | tail -n 20
[  208.986402] audit: type=1400 audit(1741663468.023:175): apparmor="DENIED" operation="mknod" class="file" profile="/us
r/bin/wsdd" name="/var/lib/libuuid/clock.txt" pid=3295 comm="python3" requested_mask="c" denied_mask="c" fsuid=1000 ouid
=1000
[  210.000181] audit: type=1400 audit(1741663469.037:176): apparmor="DENIED" operation="connect" class="file" profile="/
usr/bin/wsdd" name="/run/uuidd/request" pid=3295 comm="python3" requested_mask="w" denied_mask="w" fsuid=1000 ouid=0
[  465.403847] audit: type=1400 audit(1741663724.408:177): apparmor="DENIED" operation="open" class="file" profile="snap
-update-ns.firefox" name="/usr/local/share/" pid=3466 comm="5" requested_mask="r" denied_mask="r" fsuid=0 ouid=0
[  468.190448] audit: type=1107 audit(1741663727.194:178): pid=968 uid=101 auid=4294967295 ses=4294967295 subj=unconfine
d msg='apparmor="DENIED" operation="dbus_method_call"  bus="system" path="/org/freedesktop/timedate1" interface="org.fre
edesktop.DBus.Properties" member="GetAll" mask="send" name=":1.119" pid=3440 label="snap.firefox.firefox" peer_pid=3587
peer_label="unconfined"
               exe="/usr/bin/dbus-daemon" sauid=101 hostname=? addr=? terminal=?'
[  468.193313] audit: type=1107 audit(1741663727.197:179): pid=968 uid=101 auid=4294967295 ses=4294967295 subj=unconfine
d msg='apparmor="DENIED" operation="dbus_method_call"  bus="system" path="/org/freedesktop/timedate1" interface="org.fre
edesktop.DBus.Properties" member="GetAll" mask="send" name=":1.119" pid=3440 label="snap.firefox.firefox" peer_pid=3587
peer_label="unconfined"
               exe="/usr/bin/dbus-daemon" sauid=101 hostname=? addr=? terminal=?'
[  540.799174] workqueue: blk_mq_run_work_fn hogged CPU for >10000us 4 times, consider switching to WQ_UNBOUND
[  595.953441] my_module: loading out-of-tree module taints kernel.
[  595.953450] my_module: module verification failed: signature and/or required key missing - tainting kernel
[  595.954639] Hello, Kernel World!
[  595.954642] my_short: 10
[  595.954643] my_int: 20
[  595.954644] my_long: 30
[  595.954645] my_string: Hello
[  595.954645] my_array[0]: 1
[  595.954646] my_array[1]: 2
[  595.954647] my_array[2]: 3
[  595.954648] my_array[3]: 4
[  595.954649] my_array[4]: 5
hosseintatar@hosseintatar-VMware-Virtual-Platform:~/Desktop/Makefile$ S
```

ماکروهای module_param و MODULE_PARM_DESC :

module_paramبرای تعریف پارامترهای ماژول استفاده می‌شود. پارامترها شامل نام متغیر، نوع آن و دسترسی‌های فایل سیستم هستند.

MODULE_PARM_DESC برای ارائه توضیحات درباره پارامترها استفاده می‌شود.

نحوه استفاده:

کد را در یک فایل با نام my_module.c ذخیره میکنیم. سپس یک فایل Makefile ایجاد میکنیم تا ماژول را کامپایل کند. ماژول را با دستور insmod بارگذاری میکنیم و پارامترها را به آن منتقل میکنیم:

- sudo insmod module1.ko my_short=10 my_int=20 my_long=30 my_string="Hello" my_array=1,2,3,4,5

در نهایت لاگ‌های هسته را با دستور sudo dmesg | tail -n 20 بررسی میکنیم تا مقادیر پارامترها را ببینیم و بعد میتوان ماژول را با دستور rmmod حذف کرد:

- sudo rmmod module1

قسمت ب : یک ماژول بنویسید که در آن بتوانید اطلاعات فرایندها را از هسته دریافت کنید و مواردي مانند شناسه فرایند، میزان مصرف CPUو میزان مصرف حافظه آن را چاپ کنید (راهنمایی: درباره struct task_struct و for_each_process جستجو کنید).

جواب :

برای نوشتن یک ماژول هسته لینوکس که اطلاعات فرایندها (process) را از هسته دریافت کند و مواردی مانند شناسه فرایند(PID) ، میزان مصرف CPU و میزان مصرف حافظه (RAM) را چاپ کند، می‌توانیم از ساختار task_struct و ماکرو for_each_process استفاده کنیم. این ساختار و ماکرو به ما امکان می‌دهند تا به اطلاعات تمام فرایندهای در حال اجرا در سیستم دسترسی داشته باشیم.

*** کد این بخش به با نام ProcessInfoModule به فایل پیوست شده است ***

ساختار task_struct :

این ساختار شامل اطلاعات کامل درباره یک فرایند است. (task->comm: نام فرایند، task->pid: شناسه فرایند(PID) ، task->utime و task->stime: زمان CPU استفاده شده توسط فرایند ،task->mm: اطلاعات حافظه فرایند)

ماکرو for_each_process :

این ماکرو به ما امکان می‌دهد تا تمام فرایندهای در حال اجرا در سیستم را پیمایش کنیم.

نحوه استفاده:

کد را در یک فایل با نام process_info.c ذخیره میکنیم. یک فایل Makefile ایجاد میکنیم تا ماژول را کامپایل کند (مانند مثال قبلی) و سپس ماژول را با دستور insmod بارگذاری میکنیم:

- sudo insmod process_info.ko

در نهایت لاگ‌های هسته را با دستور sudo dmesg | tail -n 20 بررسی میکنیم تا مقادیر پارامترها را ببینیم و بعد میتوان ماژول را با دستور rmmod حذف کرد:

- sudo rmmod process_info

اجرای این کد در عکس زیر امده است:

```
hosseintatar@hosseintatar-VMware-Virtual-Platform:~/Desktop/ProcessInfoModule$ make
make -C /lib/modules/6.11.0-18-generic/build M=/home/hosseintatar/Desktop/ProcessInfoModule modules
make[1]: Entering directory '/usr/src/linux-headers-6.11.0-18-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: x86_64-linux-gnu-gcc-14 (Ubuntu 14.2.0-4ubuntu2) 14.2.0
  You are using:           gcc-14 (Ubuntu 14.2.0-4ubuntu2) 14.2.0
  CC [M]  /home/hosseintatar/Desktop/ProcessInfoModule/process_info.o
  MODPOST /home/hosseintatar/Desktop/ProcessInfoModule/Module.symvers
  CC [M]  /home/hosseintatar/Desktop/ProcessInfoModule/process_info.mod.o
  LD [M]  /home/hosseintatar/Desktop/ProcessInfoModule/process_info.ko
  BTF [M] /home/hosseintatar/Desktop/ProcessInfoModule/process_info.ko
Skipping BTF generation for /home/hosseintatar/Desktop/ProcessInfoModule/process_info.ko due to unavailability of vmlinu
x
make[1]: Leaving directory '/usr/src/linux-headers-6.11.0-18-generic'
hosseintatar@hosseintatar-VMware-Virtual-Platform:~/Desktop/ProcessInfoModule$ sudo insmod process_info.ko
[sudo] password for hosseintatar:
hosseintatar@hosseintatar-VMware-Virtual-Platform:~/Desktop/ProcessInfoModule$ sudo dmesg | tail -n 20
[  752.506009] Process: gnome-terminal- (PID: 4998)
[  752.506010] CPU Usage: 745000000
[  752.506011] Memory Usage: 54824960 bytes
[  752.506011] ---------------------------------------
[  752.506012] Process: bash (PID: 5005)
[  752.506013] CPU Usage: 27000000
[  752.506014] Memory Usage: 5963776 bytes
[  752.506015] ---------------------------------------
[  752.506015] Process: sudo (PID: 5280)
[  752.506016] CPU Usage: 24000000
[  752.506017] Memory Usage: 8110080 bytes
[  752.506018] ---------------------------------------
[  752.506019] Process: sudo (PID: 5281)
[  752.506020] CPU Usage: 1000000
[  752.506021] Memory Usage: 2674688 bytes
[  752.506022] ---------------------------------------
```

سوال دوم : یک ماژول بنویسید که در تابع ابتدایی آن، پنج عنصر struct birthday ایجاد کند و آن را پیمایش کرده و اطلاعات هر عضو را چاپ کند. همچنین در هنگام خروج نیز این لیست را به صورت برعکس پیمایش کنید و سپس هر کدام را از لیست حذف کرده و فضای آزاد شده را به هسته برگردانید.

جواب :

برای نوشتن این ماژول، از لیست‌های پیوندی (Linked Lists) در هسته لینوکس استفاده می‌کنیم. لیست‌های پیوندی در هسته لینوکس با استفاده از ساختار list_head و توابع مرتبط با آن (مانند LIST_HEAD, list_add_tail, list_for_each_entry و list_del) پیاده‌سازی می‌شوند. در این ماژول، یک لیست پیوندی از ساختار struct birthday ایجاد می‌کنیم، آن را پیمایش کرده و اطلاعات هر عضو را چاپ می‌کنیم. سپس در هنگام خروج، لیست را به صورت برعکس پیمایش کرده و هر عضو را حذف می‌کنیم.

*** کد این بخش به با نام StructBirthdayModule به فایل پیوست شده است ***

ساختار struct birthday :

این ساختار شامل سه فیلد day, month, و year است. همچنین شامل یک فیلد list از نوع struct list_head است که برای پیوند دادن این ساختار به لیست پیوندی استفاده می‌شود. لیست پیوندی با استفاده از LIST_HEAD(birthday_list) تعریف و مقداردهی اولیه می‌شود. Kmalloc برای اختصاص حافظه از هسته استفاده می‌شود و kfree رای آزاد کردن حافظه استفاده می‌شود.

توابع لیست پیوندی:

- list_add_tail: عنصر را به انتهای لیست اضافه می‌کند.
- list_for_each_entry: لیست را پیمایش می‌کند.
- list_for_each_entry_safe_reverse: لیست را به صورت برعکس و ایمن (قابل حذف) پیمایش می‌کند.
- list_del: عنصر را از لیست حذف می‌کند.

نحوه استفاده:

کد را در یک فایل با نام birthday_module.c ذخیره میکنیم. یک فایل Makefile ایجاد میکنیم تا ماژول را کامپایل کند (مانند مثال قبلی) و سپس ماژول را با دستور insmod بارگذاری میکنیم:

- sudo insmod birthday_module.ko

سپس باید ماژول را با دستور rmmod حذف کرد:

- sudo rmmod birthday_module

در نهایت لاگ‌های هسته را با دستور sudo dmesg | tail -n 30 بررسی میکنیم تا مقادیر پارامترها را ببینیم
اجرای این کد در عکس زیر امده است:

```
hosseintatar@hosseintatar-VMware-Virtual-Platform:~/Desktop/StructBirthdayModule$ sudo dmesg | tail -n 30
[  752.506019] Process: sudo (PID: 5281)
[  752.506020] CPU Usage: 1000000
[  752.506021] Memory Usage: 2674688 bytes
[  752.506022] ---------------------------------------
[  752.506022] Process: insmod (PID: 5282)
[  752.506023] CPU Usage: 4000000
[  752.506024] Memory Usage: 3682304 bytes
[  752.506025] ---------------------------------------
[  914.455503] workqueue: blk_mq_run_work_fn hogged CPU for >10000us 11 times, consider switching to WQ_UNBOUND
[  939.293156] Exiting process info module
[ 2534.018331] Starting birthday module
[ 2534.018352] Added: Day=1, Month=1, Year=2000
[ 2534.018354] Added: Day=2, Month=2, Year=2001
[ 2534.018355] Added: Day=3, Month=3, Year=2002
[ 2534.018356] Added: Day=4, Month=4, Year=2003
[ 2534.018357] Added: Day=5, Month=5, Year=2004
[ 2534.018358] Traversing the list:
[ 2534.018358] Day=1, Month=1, Year=2000
[ 2534.018359] Day=2, Month=2, Year=2001
[ 2534.018360] Day=3, Month=3, Year=2002
[ 2534.018361] Day=4, Month=4, Year=2003
[ 2534.018361] Day=5, Month=5, Year=2004
[ 2562.757955] workqueue: e1000_watchdog [e1000] hogged CPU for >10000us 4 times, consider switching to WQ_UNBOUND
[ 2871.906982] Exiting birthday module
[ 2871.906998] Traversing the list in reverse and freeing memory:
[ 2871.906999] Removing: Day=5, Month=5, Year=2004
[ 2871.907000] Removing: Day=4, Month=4, Year=2003
[ 2871.907001] Removing: Day=3, Month=3, Year=2002
[ 2871.907002] Removing: Day=2, Month=2, Year=2001
[ 2871.907003] Removing: Day=1, Month=1, Year=2000
```

سوال سوم (تشویقی) : یک مثال ساده دیگر با استفاده از rbtree یا hashtable را انجام دهید.

جواب :

*** من در حل این سوال از تابع hashtable استفاده نموده ام.

در این مثال، یک ماژول ساده می‌نویسیم که از hashtable برای ذخیره و مدیریت داده‌ها استفاده می‌کند. این ماژول یک hashtable ایجاد می‌کند، چند عنصر به آن اضافه می‌کند، عناصر را جستجو می‌کند و در نهایت hashtable را پاک می‌کند.

*** کد این بخش به با نام HashTableModule به فایل پیوست شده است ***

ساختار struct my_data:

این ساختار شامل دو فیلد key و value است. همچنین شامل یک فیلد node از نوع struct hlist_node است که برای پیوند دادن این ساختار به hashtable استفاده می‌شود. kmalloc برای اختصاص حافظه از هسته استفاده می‌شود kfree برای آزاد کردن حافظه استفاده می‌شود.

تعریف و توابع hashtable:

DEFINE_HASHTABLE(my_hashtable, HASHTABLE_SIZE) یک hashtable با اندازه HASHTABLE_SIZE تعریف می‌کند.
hash_add عنصر را به hashtable اضافه می‌کند.
hash_for_each_possible عناصر با کلید مشابه را پیمایش می‌کند.
hash_for_each_safe: hashtable را به صورت ایمن (قابل حذف) پیمایش می‌کند.
hash_del عنصر را از hashtable حذف می‌کند.

نحوه استفاده:

کد را در یک فایل با نام hashtable_module.c ذخیره میکنیم. یک فایل Makefile ایجاد میکنیم تا ماژول را کامپایل کند (مانند مثال قبلی) و سپس ماژول را با دستور insmod بارگذاری میکنیم:

- sudo insmod hashtable_module.ko

سپس باید ماژول را با دستور rmmod حذف کرد:

- sudo rmmod hashtable_module

در نهایت لاگ‌های هسته را با دستور sudo dmesg | tail -n 13 بررسی میکنیم تا مقادیر پارامترها را ببینیم
اجرای این کد در عکس زیر امده است:

```
hosseintatar@hosseintatar-VMware-Virtual-Platform:~/Desktop/HashTableModule$ sudo dmesg | tail -n 13
[ 4345.263115] Starting hashtable module
[ 4345.263131] Added: Key=0, Value=0
[ 4345.263132] Added: Key=1, Value=100
[ 4345.263133] Added: Key=2, Value=200
[ 4345.263134] Added: Key=3, Value=300
[ 4345.263134] Added: Key=4, Value=400
[ 4345.263135] Found: Key=3, Value=300
[ 4359.110351] Exiting hashtable module
[ 4359.110355] Removing: Key=0, Value=0
[ 4359.110357] Removing: Key=3, Value=300
[ 4359.110358] Removing: Key=1, Value=100
[ 4359.110359] Removing: Key=4, Value=400
[ 4359.110359] Removing: Key=2, Value=200
hosseintatar@hosseintatar-VMware-Virtual-Platform:~/Desktop/HashTableModule$
```