**Проектирование ОС и компонентов**

Системные вызовы

**Теоретическая часть**

Системный вызов в программировании и вычислительной технике — обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции. Современные операционные системы (ОС) предусматривают разделение времени между выполняющимися вычислительными процессами (многозадачность) и разделение полномочий, препятствующее обращению исполняемых программ к данным других программ и оборудованию. Ядро ОС исполняется в привилегированном режиме работы процессора. Для выполнения межпроцессной операции или операции, требующей доступа к оборудованию, программа обращается к ядру, которое, в зависимости от полномочий вызывающего процесса, исполняет либо отказывает в исполнении такого вызова. С точки зрения программиста, системный вызов обычно выглядит как вызов подпрограммы или функции из системной библиотеки. Однако системный вызов, как частный случай вызова такой функции или подпрограммы, следует отличать от более общего обращения к системной библиотеке, поскольку последнее может и не требовать выполнения привилегированных операций.

Режим ядра

Когда процессор находится в режиме ядра, код может быть выполнен с любым адресом памяти и любым аппаратным ресурсом. Следовательно, режим ядра - очень привилегированный и мощный режим. Если программа выйдет из строя в режиме ядра, вся система будет остановлена.

Пользовательский режим

Когда CPU находится в пользовательском режиме, программы не имеют прямого доступа к памяти и аппаратным ресурсам. В режиме пользователя, если какая-либо программа выходит из строя, останавливается только эта конкретная программа. Это означает, что система будет находиться в безопасном состоянии, даже если программа в пользовательском режиме сработает.

Следовательно, большинство программ в ОС запускаются в пользовательском режиме.

**Системные вызовы Syslog, Ptrace, Uname, Sysinfo**

**Syslog**

Краткий обзор

|  |
| --- |
| int syslog(int type, char \*bufp, int len);  /\* No wrapper provided in glibc \*/  /\* The glibc interface \*/  #include <sys/klog.h>  int klogctl(int type, char \*bufp, int len); |

Ядро имеет циклический буфер длиной LOG\_BUF\_LEN, в котором сообщения

       Заданные в качестве аргументов функции ядра printk (), сохраняются (

       Меньше их loglevel). В ранних ядрах значение LOG\_BUF\_LEN имело значение

       4096; Из ядра 1.3.54 это было 8192; Из ядра 2.1.113 это было

       16384; Поскольку 2.4.23 / 2.6 это параметр конфигурации ядра. В

       Последние ядра размер может быть запрошен с типом команды 10.

* Вызов syslog ожидает, пока этот буфер журнала ядра не будет

       Непустым, а затем считывает не более len байтов в буфер buf. Это

       Возвращает количество прочитанных байтов. Байты, считанные из журнала, исчезают

       Из буфера журнала: информация может быть прочитана только один раз. Это

       Функция, выполняемая ядром при чтении пользовательской программы

       / Proc / kmsg.

* В вызове syslog) будут прочитаны последние len байты из журнала

       Буфера (неразрушающе), но не будет читать больше, чем было записано в

       Буфер с последней команды «clear ring buffer» (которая не

       Полностью очистить буфер). Он возвращает количество прочитанных байтов.

* Вызов syslog делает то же самое, но также выполняет

       Команда «clear ring buffer».

* Вызов syslog выполняет только «очищающий буфер звонка»,

       команда. (В каждом вызове, где buf или len отображается как «фиктивный», значение

       Аргумента игнорируется вызовом.)

* Вызов syslog устанавливает минимальный уровень журнала консоли,

       Так что на консоль не печатаются сообщения.

* Syslog вызова) устанавливает уровень журнала консоли по умолчанию,

       Так что сообщения печатаются на консоль.

* Вызов syslog) устанавливает уровень журнала консоли на уровень, который должен быть целым числом от 1 до 8 (включительно).

       Для подробностей.

* В вызове syslog возвращается количество байтов в настоящее время. Доступный для чтения в буфере журнала ядра.
* Вызов syslog возвращает общий размер ядра. Буфер журнала.

ОШИБКИ

        EINVAL Плохие аргументы (например, плохой тип, или для типа 2, 3 или 4, buf -

NULL или len меньше нуля; Или для типа 8,

На стороне от 1 до 8).

        EPERM Была сделана попытка изменить console\_loglevel или очистить кера-

Буфера кольцевого буфера сообщений процессом без достаточных

Lege (точнее: без возможности CAP\_SYS\_ADMIN).

ERESTARTSYS

Сигнал системы был прерван сигналом; Ничего не было прочитано.

(Это можно увидеть только во время трассировки).

        ENOSYS Этот системный вызов syslog () недоступен, поскольку ядро

Был скомпилирован с параметром конфигурации ядра CONFIG\_PRINTK

отключен.

Исходный код:

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | /\*\* Test program for the syslog() system call.  \*/  #include "../../config.h"  #include <stdio.h>  #if defined(HAVE\_SYS\_KLOG\_H)  #include <sys/klog.h>  #endif  int main(int argc, char\*\* argv)  {  int number\_of\_unread\_characters;  #if defined HAVE\_KLOGCTL  number\_of\_unread\_characters = klogctl(9, 0, 0);  #endif  fprintf(stderr, "Done.\n");  return 0 \* number\_of\_unread\_characters;  } |  | |

**Список используемых источников:**