


Программирование под Linux

Дубров Денис Владимирович
Ведущий инженер-программист, к.ф.-м.н.

denis.dubrov@harman.com

Определение

Переносимый интерфейс операционных систем Unix: (Portable Operating System Interface for Unix, POSIX) — семейство стандартов IEEE, направленных на совместимость между ОС: IEEE 1003.1-2008.

ОС, совместимые с POSIX

Полностью

- HP-UX
- macOS
- Solaris
- AIX
- QNX
- ...

В основном

- FreeBSD
- GNU/Linux
- MINIX
- ...

В Windows

- Cygwin
- MinGW
- Microsoft POSIX Subsystem
- ...

Прикладные программные интерфейсы

Стандарты API

- Windows API (Win32 API);
- POSIX.

Пример (вход/выход из режима ядра, X86)

	Ранние версии	Intel Pentium II/Linux 2.6
Вход	`int $0x80`	`sysenter`
Выход	`iret`	`sysexit`

Вызов функции API

Пример (C)

```c
if (MessageBeep(MB_ICONHAND))
  /* ... */;
```

Пример (Assembler X86)

```
    push    31h
    push    10h  ; MB_ICONHAND
    mov     eax, 1143h
    push    eax
    push    @f
    mov     edx, esp
    sysenter
@@: add     esp, 12
```

Создание процесса, Windows API

Функция создания процесса

```
BOOL WINAPI CreateProcess(
  _In_opt_    LPCTSTR               lpApplicationName,
  _Inout_opt_ LPTSTR                lpCommandLine,
  _In_opt_    LPSECURITY_ATTRIBUTES lpProcessAttributes,
  _In_opt_    LPSECURITY_ATTRIBUTES lpThreadAttributes,
  _In_        BOOL                  bInheritHandles,
  _In_        DWORD                 dwCreationFlags,
  _In_opt_    LPVOID                lpEnvironment,
  _In_opt_    LPCTSTR               lpCurrentDirectory,
  _In_        LPSTARTUPINFO         lpStartupInfo,
  _Out_       LPPROCESS_INFORMATION lpProcessInformation
);
```

Создание процесса, POSIX

Функции создания процесса (<unistd.h>)

```
pid_t fork(void);

int execve(
  const char *file, char *const argv[], char *const envp[]);

int execl(const char *path, const char *arg, ...);
int execlp(const char *file, const char *arg, ...);
int execle(
  const char *path, const char *arg, ..., char *const envp[]);
int execv(const char *path, char *const argv[]);
int execvp(const char *file, char *const argv[]);
```

Завершение процесса

Функции завершения процесса, POSIX (<stdlib.h>, <unistd.h>)

```
void  exit(int status);    /* EXIT_SUCCESS, EXIT_FAILURE */
void _Exit(int status);    /* C99 */
void _exit(int status);
```

Принудительное завершение процесса

Функция завершения процесса, POSIX (<signal.h>)

```
int kill(pid_t pid, int sig);
```

Определение

Сигнал: средство межпроцессного взаимодействия в POSIX-совместимой ОС. Представляет собой асинхронное сообщение процессу или потоку, прерывая его на неатомарной операции. При появлении сигнала в контексте процесса-получателя вызывается **обработчик сигнала**.

Виды сигналов

Имя	№	Клавиши	Перехват	Значение
SIGKILL	9	—	✗	немедленное завершение
SIGINT	2	Ctrl + C	✓	прерывание
SIGTSTOP		—	✗	временный останов
SIGTSTP		Ctrl + Z	✓	временный останов с терминала
SIGCONT		—	✓	продолжение после SIGTSTOP, SIGTSTP
SIGSEGV		—	✓	неправильный доступ к памяти
SIGILL		—	✓	неправильная машинная инструкция
SIGFPE		—	✓	неправильная мат. операция
SIGPIPE		—	✓	запись в канал, из кот. не читают

Таблица: некоторые часто используемые сигналы POSIX

Ожидание дочернего процесса

Функции ожидания процесса (<sys/wait.h>)

```
pid_t wait(int *pnStatus);
pid_t waitpid(pid_t pid, int *pnStatus, int nOptions);
```

Проверка	Дополнительная информация
WIFEXITED(nStatus)	WEXITSTATUS(nStatus)
WIFSIGNALED(nStatus)	WTERMSIG(nStatus), WCOREDUMP(nStatus)
WIFSTOPPED(nStatus)	WSTOPSIG(nStatus)
WIFCONTINUED(nStatus)	

Таблица: макросы проверки состояния завершения процесса

Создание процесса

Пример

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>

int main(int nArgC, char *apszArgV[])
{
  pid_t pid;
  int nStatus;
```

Создание процесса (продолжение)

Пример (продолжение)

```
if (nArgC < 2)
{
  printf("Usage: %s command, [arg1 [arg2]...]\n", apszArgV[0]);
  return EXIT_FAILURE;
}
//
printf("Starting %s...\n", apszArgV[1]);
pid = fork();
```

Создание процесса (продолжение)

Пример (продолжение)

```
if (pid == 0)
{
  execvp(
    apszArgV[1], &apszArgV[1]);
  perror("execvp");
  return EXIT_FAILURE;
}
```

Пример (продолжение)

```
else
{
  if (wait(&nStatus) == -1)
  {
    perror("wait");
    return EXIT_FAILURE;
  }
```

Создание процесса (продолжение)

Пример (продолжение)

```
    if (WIFEXITED(nStatus))
      printf(
        "Child terminated normally with exit code %i\n",
        WEXITSTATUS(nStatus));
    if (WIFSIGNALED(nStatus))
    {
      printf(
        "Child was terminated by a signal #%i\n", WTERMSIG(nStatus));
#ifdef WCOREDUMP
      if (WCOREDUMP(nStatus))
        printf("Child dumped a core\n");
#endif    // WCOREDUMP
```

Создание процесса (окончание)

Пример (окончание)

```
    }
    if (WIFSTOPPED(nStatus))
      printf(
        "Child was stopped by a signal #%i\n",
        WSTOPSIG(nStatus));
  }    // if (pid == 0) (else)
  //
  return EXIT_SUCCESS;
}    // main()
```

Работа создания процесса

Пример (работа программы)

```
$ gcc -o exec fork_posix.c
$ ./exec ./exec ls -al
Starting ./exec...
Starting ls...
total 24
drwxr-xr-x 2 dubrov dubrov 4096 Sep 16 01:11 .
drwxrwxr-x 3 dubrov dubrov 4096 Sep 16 01:10 ..
-rwxrwxr-x 1 dubrov dubrov 8772 Sep 16 01:11 exec
-rw-r--r-- 1 dubrov dubrov 1300 Sep 16 01:11 fork_posix.c
Child terminated normally with exit code 0
Child terminated normally with exit code 0
```

Дерево процессов

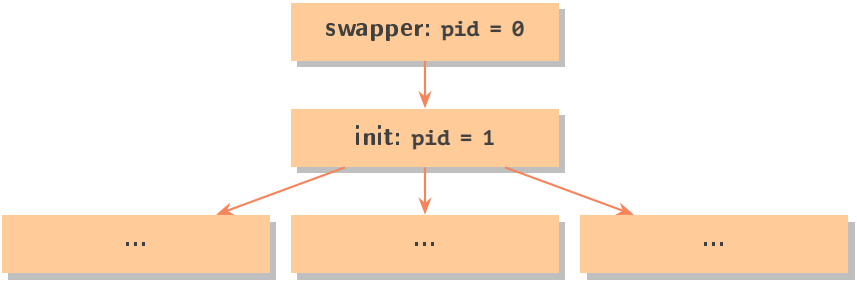

Рис.: дерево процессов в Linux и других ОС

Вывод дерева процессов в ОС Linux

Пример

```
$ pstree -p -A
init(1)-+-NetworkManager(506)-+-dhclient(591)
        |                     |-{NetworkManager}(518)
        |                     `-{NetworkManager}(592)
        |-accounts-daemon(1077)---{accounts-daemo}(1078)
        |-acpid(755)
        |-atd(762)
        |-avahi-daemon(500)---avahi-daemon(507)
        |-bamfdaemon(1920)---{bamfdaemon}(1928)
        |-bluetoothd(840)
        |-colord(962)-+-{colord}(974)
        |             `-{colord}(1046)
        ...
```

Работа с каналами

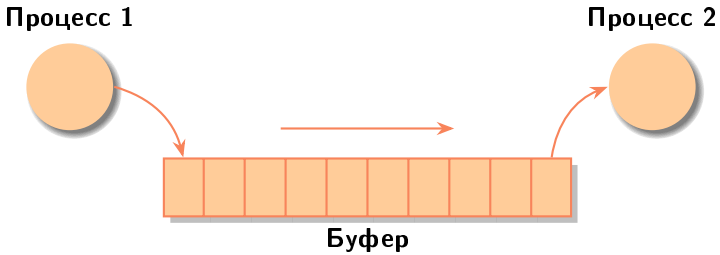

Рис.: принцип обмена данными при помощи канала

Каналы POSIX

unistd.h

```
int     pipe(int anFD[2]);     // anFD[0] – чтение,
                               // anFD[1] – запись

int     dup2(int nOldFD, int nNewFD);
ssize_t read(int nFD, void *pvBuf, size_t uCount);
ssize_t write(int nFD, const void *pcvBuf, size_t uCount);
int     close(int nFD);
```

Использование неименованных каналов

Пример

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

void check(
  int nResult, const char *pcszErrMsg)
{
  if (nResult == -1)
  {
    perror(pcszErrMsg);
    exit(EXIT_FAILURE);
  }
}
```

Пример (продолжение)

```
void createPipe(int anFD[2])
{
  const int cnResult = pipe(anFD);
  check(cnResult, "pipe");
}
```

Использование неименованных каналов (продолжение)

Пример (продолжение)

```
int main()
{
  const size_t cuSize = sizeof (int);
  int anPipeToChild[2], anPipeToParent[2];
  createPipe(anPipeToChild);
  createPipe(anPipeToParent);
  const pid_t cpid = fork();
  check(cpid, "fork");
  if (cpid == 0)
  {
```

Использование неименованных каналов (продолжение)

Пример (продолжение)

```
// Child
close(anPipeToChild[1]);
close(anPipeToParent[0]);
int nData;
ssize_t nRead;
do
{
  nRead = read(anPipeToChild[0], &nData, cuSize);
  check(nRead, "read");
  ++ nData;
  const ssize_t cnWritten = write(anPipeToParent[1], &nData, cuSize);
  check(cnWritten, "write");
```

Пример (продолжение)

```
    }
    while (nRead == cuSize);
}
else    // if (cpid == 0)
{
  // Parent
  close(anPipeToChild[0]);
  close(anPipeToParent[1]);
  int nData = 0;
```

Использование неименованных каналов (окончание)

Пример (окончание)

```
    for (int i = 0; i < 500; ++ i)
    {
      const ssize_t cnWritten = write(anPipeToChild[1], &nData, cuSize);
      check(cnWritten, "wirte");
      const ssize_t cnRead = read(anPipeToParent[0], &nData, cuSize);
      check(cnRead, "read");
      std::cout << nData << ' ';
    }
    std::cout << std::endl;
  }    // if (cpid == 0) (else)
}    // main()
```

Использование каналов в POSIX

Пример

```
$ ls | more
```

Реализация bash

1. `pipe(anFD)`
2. `fork()` (2 раза)
3. `close(anFD[i])` (2 раза)

Реализация ls

1. `dup2(anFD[1], 1)`
2. `close(anFD[i])` (2 раза)
3. `execve( "ls", argv, envp)`

Реализация more

1. `dup2(anFD[0], 0)`
2. `close(anFD[i])` (2 раза)
3. `execve( "more", argv, envp)`

Высокоуровневое управление каналами в POSIX

stdio.h

```
FILE * popen(const char *pcszCommand, const char *pcszType);
int    pclose(FILE *stream);

FILE * fdopen(int nFD, const char *pcszMode);
size_t fread(
  void *pvBuf, size_t uSize, size_t uCount, FILE *stream);
size_t fwrite(
  const void *pcvBuf, size_t uSize, size_t uCount, FILE *stream);
int    feof(FILE *stream);
int    fclose(FILE *stream);
```

Реализация popen()

Родительский процесс

1. `pipe(anFD)`
2. `fork()`
3. **если** `pcszType` $\sim$ `"r"`, **то**
   `close(anFD[1])`
   иначе // *pcszType* $\sim$ *"w"*
   `close(anFD[0])`
4. **если** `pcszType` $\sim$ `"r"`, **то**
   вернуть `fdopen(anFD[0], pcszType)`
   иначе // *pcszType* $\sim$ *"w"*
   вернуть `fdopen(anFD[1], pcszType)`

Реализация popen() (окончание)

Дочерний процесс

1. **если** pcszType $\sim$ "r", **то**
   | dup2(anFD[1], 1)
   иначе // *pcszType* $\sim$ *"w"*
   ⌊ dup2(anFD[0], 0)
2. close(anFD[i]) (2 раза)
3. execve(pcszCommand, argv, envp)

Реализация `pclose()`

Родительский процесс

1. `wait()` для дочернего процесса.

Именованные каналы POSIX

POSIX `mkfifo()`

```
#include <sys/types.h>
#include <sys/stat.h>

int mkfifo(const char *pcszPathName, mode_t ulMode);
```

Пример (fifo_server.c)

```c
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Пример (fifo_server.c, продолжение)

```c
#define PIPE_NAME_1 "to-server"
#define PIPE_NAME_2 "from-server"
#define RESPONSE_ST "Response from server"
#define BUFFER_SIZE 50

// void check(...) ...
```

Каналы POSIX (продолжение)

Пример (`fifo_server.c`, продолжение)

```
int main()
{
  int nResult, nFD1, nFD2;
  ssize_t i, nLength;
  char achBuffer[BUFFER_SIZE];
  const char *pcszResponse = RESPONSE_ST;
  //
  printf("Starting server...\n");
  nResult = unlink(PIPE_NAME_1);    // May fail
  //
  nResult = unlink(PIPE_NAME_2);    // May fail
  //
```

Каналы POSIX (продолжение)

Пример (`fifo_server.c`, продолжение)

```
nResult = mkfifo(PIPE_NAME_1, S_IWUSR | S_IRUSR);
check(nResult, "server mkfifo 1");
//
nResult = mkfifo(PIPE_NAME_2, S_IWUSR | S_IRUSR);
check(nResult, "server mkfifo 2");
//
printf("Opening...\n");
nFD1 = open(PIPE_NAME_1, O_WRONLY);
check(nResult, "server open 1");
//
nFD2 = open(PIPE_NAME_2, O_RDONLY);
check(nResult, "server open 2");
```

Каналы POSIX (продолжение)

Пример (`fifo_server.c`, окончание)

```
//
do
{
  nLength = read(nFD2, achBuffer, BUFFER_SIZE);
  check(nLength, "server read");
  for (i = 0; i < nLength; ++ i)
    putchar(achBuffer[i]);
}
while (nLength == BUFFER_SIZE);
putchar('\n');
//
```

Каналы POSIX (продолжение)

Пример (`fifo_server.c`, окончание)

```
  printf("Writing server...\n");
  nLength = write(nFD1, pcszResponse, strlen(pcszResponse));
  check(nLength, "server write");
  printf("Wrote server!\n");
  //
  close(nFD1);
  close(nFD2);
}    // main()
```

Каналы POSIX (продолжение)

Пример (`fifo_client.c`)

```c
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Пример (`fifo_client.c`, продолжение)

```c
#define PIPE_NAME_1 "to-server"
#define PIPE_NAME_2 "from-server"
#define RESPONSE_ST "Response from client"
#define BUFFER_SIZE 50

// void check(...) ...
```

Каналы POSIX (продолжение)

Пример (`fifo_client.c`, продолжение)

```
int main()
{
  int nResult, nFD1, nFD2;
  ssize_t i, nLength;
  char achBuffer[BUFFER_SIZE];
  const char *pcszResponse = RESPONSE_ST;
  //
  printf("Starting client...\n");
  //
```

Каналы POSIX (продолжение)

Пример (`fifo_client.c`, продолжение)

```
printf("Opening...\n");
nFD2 = open(PIPE_NAME_1, O_RDONLY);
check(nFD2, "client open 2");
//
nFD1 = open(PIPE_NAME_2, O_WRONLY);
check(nFD1, "client open 1");
printf("Opened!\n");
//
```

Каналы POSIX (продолжение)

Пример (`fifo_client.c`, продолжение)

```
printf("Writing client...\n");
nLength = write(nFD1, pcszResponse, strlen(pcszResponse));
check(nLength, "client write");
printf("Wrote client!\n");
//
```

Каналы POSIX (окончание)

Пример (fifo_client.c, продолжение)

```
do
{
  nLength = read(nFD2, achBuffer, BUFFER_SIZE);
  check(nLength, "client read");
  for (i = 0; i < nLength; ++ i)
    putchar(achBuffer[i]);
}
while (nLength == BUFFER_SIZE);
putchar('\n');
```

Пример (окончание)

```
  //
  close(nFD1);
  close(nFD2);
}    // main()
```

Создание сокета домена Unix

Пример (socket(), <sys/socket.h>, <sys/un.h>, fcntl(), <unistd.h>, <fcntl.h>)

```
int nFD = socket(AF_UNIX, SOCK_STREAM, 0);
check(nFD, "server socket");
//
int nFlags = fcntl(nFD, F_GETFL, 0);
check(nFlags, "server fcntl get");
nFlags = fcntl(nFD, F_SETFL, nFlags | O_NONBLOCK);
check(nFlags, "server fcntl set");
```

Привязка сокета к имени, перевод в пассивный режим

Пример (bind(), listen(), <sys/socket.h>, <sys/types.h>)

```
struct sockaddr_un addr;
memset(&addr, 0, sizeof (addr));
addr.sun_family = AF_UNIX;
strncpy(addr.sun_path, g_cszSocketPath, sizeof (addr.sun_path) - 1);
unlink(g_cszSocketPath);
//
int nRet = bind(nFD, (struct sockaddr *) &addr, sizeof (addr));
check(nRet, "server bind");
//
nRet = listen(nFD, SOMAXCONN);
check(nRet, "server listen");
```

Извлечение первого запроса на соединение

Пример (`accept()`, `<sys/socket.h>`, `<sys/types.h>`)

```
struct sockaddr_un addr_remote;
socklen_t uSockLen;
int nConnection = accept(
  nFD, (struct sockaddr *) &addr_remote, &uSockLen);
if (nConnection < 0)
{
  if (errno == EAGAIN || errno == EWOULDBLOCK)
    continue;
  //
  perror("server accept");
  continue;
}
```

Запрос клиента на соединение

Пример (connect(), <sys/socket.h>, <sys/types.h>)

```
int nFD = socket(AF_UNIX, SOCK_STREAM, 0);
check(nFD, "client socket");
//
struct sockaddr_un addr;
// ... same as in the server ...
int nRet = connect(nFD, (struct sockaddr *) &addr, sizeof (addr));
check(nRet, "client connect");
```

Запись в поток данных

Пример (`send()`, `<sys/socket.h>`, `<sys/types.h>`)

```
ssize_t nSize = send(nFD, pcszBuffer, cuLength, 0);
check(nSize, "client send");
if (nSize < cuLength)
   printf("Partially written\n");
```

Чтение потока данных

Пример (`recv()`, `<sys/socket.h>`, `<sys/types.h>`)

```
do
{
  nSize = recv(nConnection, szBuffer, sizeof (szBuffer), MSG_WAITALL);
  check(nSize, "server recv");
  // ... use szBuffer ...
}
while (nSize > 0);
close(nConnection);
```

Разделяемая память POSIX

POSIX `shmget()`, `<sys/types.h>`, `<sys/shm.h>`

```
int    shmget(key_t nKey, int nSize, int nShmFlg);
```

nKey	nShmFlg
IPC_PRIVATE	IPC_CREAT
	IPC_EXCL
	младшие 9 бит

Таблица: возможные значения флагов параметров функции `shmget()`

Разделяемая память POSIX (продолжение)

POSIX shmat(), shmdt()

```
#include <sys/types.h>
#include <sys/shm.h>

void *shmat(int nShmId, const void *pvShmAddr, int nShmFlg);
int   shmdt(const void *pvShmAddr);
```

nShmFlg
SHM_RND (SHMLBA)
SHM_RDONLY

Таблица: возможные значения флагов параметров функции shmat()

Разделяемая память POSIX (окончание)

Пример

```
const key_t g_cKey = 1917;
// ...
int nShmId = shmget(g_cKey, sizeof (struct connect), IPC_CREAT | 0644);
check(nShmId, "shmget");
//
struct connect *pConnect = (struct connect *) shmat(nShmId, NULL, 0);
// ...
shmdt(pConnect);
```

Решение проблемы дублирования ключей

Варианты

- Использование в качестве ключа константы `IPC_PRIVATE`.
- Генерирование ключа при помощи функции `ftok()`.

POSIX `ftok()`

```
#include <sys/types.h>
#include <sys/ipc.h>

key_t ftok(const char *pcszPathName, int nProjId);
```




АКАДЕМИЯ АЙТИ

Спасибо за внимание!

Центральный офис:
Москва, Варшавское шоссе 47,
корп.4, 10 этаж
Тел: +7 (495) 662-7894, 662-7895
Факс: +7(495) 974-7990
e-mail: academy@it.ru

www.academy.it.ru