| **Наз-е** | **Прототип** | **Описание** | **Что возвращает** | **Пример** |
| --- | --- | --- | --- | --- |
| write |  |  |  |  |
| signal | typedef void signalfunction(int);  signalfunction \*signal(int signr,  signalfunction \*sighandler);  void (\*signal(int sign, void (\*funcptr)(int)))(int); | определяет способ обработки сигналов  *sign* - Номер сигнала, который функция будет обрабатывать  Есть три способа, обработки сигнала программой (способ обработки указывается параметром *funcptr*):   * Обработка сигналов по умолчанию SIG\_DFL: сигнал обрабатывается, по умолчанию, событием для данного сигнала. * Игнорировать сигнал SIG\_IGN: сигнал игнорируется и выполнение кода будет продолжаться, даже если это бессмысленно. * Функция обработчик: специфическая функция, определена для обработки сигналов. | Функция возвращает указатель на функцию, которая была перед этим задана в качестве обработчика.  Если функция отработала с ошибкой, то она вернёт значение SIGERR. |  |
| sigemptyset | int sigemptyset(sigset\_t \*set); | инициализирует набор сигналов, задаваемый set, пустым значением, то есть все сигналы исключены из набора | При успешном выполнении функции возвращают 0 и -1 при ошибке. | #include <stdio.h>  #include <stdlib.h>  #include <signal.h>  void term\_handler(int i)  {  printf ("Terminating\n");  exit(EXIT\_SUCCESS);  }  int main(int argc, char \*\* argv) {  struct sigaction sa;  sigset\_t newset;  sigemptyset(&newset);  **/\*инициализация наборов сигналов пустыми значениями\*/**  sigaddset(&newset, SIGHUP);  **/\*добавление значения SIGHUP в пустой набор\*/**  sigprocmask(SIG\_BLOCK, &newset, 0);  sa.sa\_handler = term\_handler;  sigaction(SIGTERM, &sa, 0); /\****устанавливаем обработчик сигнала SIGTERM - 0 - игнорируем сигнал \*/***  printf("My pid is %i\n", getpid());  printf("Waiting...\n");  while(1)  sleep(1);  return EXIT\_FAILURE;  } |
| sigaddset | int sigaddset(sigset\_t \*set, int signum); | добавляет сигнал signum из set. |
| sigaction | int sigaction(int signum, const struct sigaction \*act,  struct sigaction \*oldact); | используется для изменения выполняемого процессом действия при получении определённого сигнала.  В signum указывается сигнал; может принимать значение любого корректного сигнала за исключением SIGKILL и SIGSTOP   * Если значение act не равно NULL, то устанавливается новое действие для сигнала signum из act. * Если значение oldact не равно NULL, то предыдущее действие записывается в oldact. | При успешном выполнении sigaction() возвращается 0; при ошибке возвращается -1, а в errno содержится код ошибки. |
| kill | int kill(pid\_t pid, int sig); | отправляет сигнал.  *sig* - является либо именем сигнала, либо 0. В последнем случае сигнал не посылается, но ядро все равно осуществляет проверку ошибок. В частности, это правильный способ проверки существования данного процесса или группы, а также проверки того, что у вас есть разрешение на передачу сигналов процессу или группе процессов | возвращает 0 в случае успеха и -1 при ошибке; | kill(getpid(), sig); |
| getpid | pid\_t getpid(void); | возвращает идентификатор ID текущего процесса. |  |
| malloc |  |  |  |  |
| free |  |  |  |  |
| pause | int pause(void); | вызывающий процесс (или подзадача) приостанавливается до тех пор, пока не получит сигнал. Данный сигнал либо остановит процесс, либо заставит его вызвать функцию обработки этого сигнала. | возвращается только тогда, когда сигнал был перехвачен и произошел возврат из функции обработки сигнала. В этом случае она возвращает -1, а значение переменной errno становится равным EINTR. |  |
| sleep |  |  |  |  |
| usleep | int usleep (useconds\_t usec); | Приостанавливает работу потока, в котором она была вызвана, на указанное в аргументе время в микросекундах или до поступления сигнала по которому вызывается функция обработки сигналов или программа завершает свою работу. | 0 - если истекло время приостановки, указанное в аргументе функции,  -1 - если приостановка была прервана сигналом |  |
| exit |  |  |  |  |

<https://wm-help.net/lib/b/book/2075737573/234> - kill

<https://learnc.info/c/signals.html> - about signals

<http://citforum.ru/programming/unix/signals/> - great topic about signal’s functions

<https://www.opennet.ru/docs/RUS/linux_parallel/node10.html> - plus one signal topic

<http://ru.manpages.org/sigemptyset/3> - man for signal’s functions

<http://ru.manpages.org/signal/7> - about signals with numbers of ones

<https://habr.com/ru/post/122823/> - IMPORTANT