1) <https://www.opennet.ru/docs/RUS/linux_parallel/node7.html>

2) <https://www.opennet.ru/docs/RUS/linux_parallel/node8.html>

3) <https://acm.bsu.by/wiki/Unix2019b/%D0%A1%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB%D1%8B>

1. Создание процессов в Linux выполняется с помощью системного вызова fork(). Он создает копию текущего процесса, которая становится дочерним процессом. Пример использования:

pid\_t fork(void);

2. Для запуска новой программы в процессе используется системный вызов exec(). Он обновляет текущий процесс с помощью новой программы, замещая его. Примеры использования:

int execl(char \*name, char \*arg0, ... /\*NULL\*/);

int execv(char \*name, char \*argv[]);

int execle(char \*name, char \*arg0, ... /\*,NULL,

char \*envp[]\*/);

int execve(char \*name, char \*arv[], char \*envp[]);

int execlp(char \*name, char \*arg0, ... /\*NULL\*/);

int execvp(char \*name, char \*argv[]);

3. Для отправки сигналов другим процессам используется функция kill(). Она посылает сигнал указанному процессу или группе процессов. Пример использования:

kill(pid, SIGTERM);

4. Для обработки сигналов в Linux существует механизм сигналов. Программы могут зарегистрировать обработчики сигналов для определенных сигналов. Для установки обработчика сигнала используется функция signal(). Пример:

void(\*signal(int signr, void(\*sighandler)(int)))(int);

5. В Linux также существуют пользовательские сигналы, которые программы могут использовать для своих нужд. Они имеют номера в диапазоне от SIGUSR1 до SIGUSR2. Пример использования:

void sig\_handler(int signum) {

if (signum == SIGUSR1) {

// пользовательский сигнал 1

} else if (signum == SIGUSR2) {

// пользовательский сигнал 2

}

}

signal(SIGUSR1, sig\_handler);