**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №4**

**по дисциплине «Операционные системы»**

**Тема: Системное программирование в ОС семейства Unix**

| Студентка гр. 2384 |  | Соц Е.А. |
| --- | --- | --- |
| Преподаватель |  | Душутина Е.В. |

Санкт-Петербург

2024

## **Цель работы.**

Целью данной работы является изучение основных принципов управления процессами и потоками в операционных системах.

## **Задание.**

*Взаимодействие родственных процессов*

13.1. Изменяя длительности выполнения процессов и параметры системных вызовов, рассмотрите 3 ситуации и получите соответствующие таблицы процессов:

а) процесс-отец запускает процесс-сын и ожидает его завершения;

б) процесс-отец запускает процесс-сын и, не ожидая его завершения, завершает свое выполнение. Зафиксируйте изменение родительского идентификатора процесса-сына;

в) процесс-отец запускает процесс-сын и не ожидает его завершения; процесс-сын завершает свое выполнение. Зафиксируйте появление процесса-зомби, для этого включите команду ps в программу father.c

13.2. Перенаправьте вывод не только на терминал, но и в файл. Организуйте программу многопроцессного функционирования так, чтобы результатом ее работы была демонстрация всех трех ситуаций с отображением в итоговом файле.

*Управление процессами посредством сигналов*

13.1. С помощью команды kill -l ознакомьтесь с перечнем сигналов, поддерживаемых процессами.

Ознакомьтесь с системными вызовами kill(2), signal(2).

Подготовьте программы следующего содержания:

а.) процесс father порождает процессы son1, son2, son3 и запускает на исполнение программные коды из соответствующих исполнительных файлов;

б.) далее родительский процесс осуществляет управление потомками, для этого он генерирует сигнал каждому пользовательскому процессу;

в.) в пользовательских процессах-потомках необходимо обеспечить: для son1 - реакцию на сигнал по умолчанию;

для son2 - реакцию игнорирования;

для son3 - перехватывание и обработку сигнала.

Сформируйте файл-проект из четырех файлов, откомпилируйте, запустите программу.

Проанализируйте таблицу процессов до и после посылки сигналов с помощью системного вызова system("ps -s >> file").

Обратите внимание на реакцию, устанавливаемую для последнего потомка.

13.2. Организуйте посылку сигналов любым двум процессам, находящимся в разных состояниях: активном и пассивном, фиксируя моменты посылки и приема каждого сигнала с точностью до секунды. Приведите результаты в файле результатов.

14. Запустите в фоновом режиме несколько утилит, например:

cat \*.c > myprog & lpr myprog & lpr intro&

Воспользуйтесь командой jobs для анализа списка заданий и очередности их выполнения.

Позаботьтесь об уведомлении о завершении одного из заданий с помощью команды notify. Аргументом команды является номер задания.

Верните невыполненные задания в приоритетный режим командой fg. Например: fg %3

Отмените одно из невыполненных заданий.

15. Ознакомьтесь с выполнением команды и системного вызова nice(1) и getpriority(2).

Приведите примеры их использования в приложении. Определите границы приоритетов (создайте для этого программу). Есть ли разница в приоритетах для системных и пользовательских процессов, используются ли приоритеты реального времени? Каков пользовательский приоритет для запуска приложений из shell? Все ответы подкрепляйте экспериментально.

16. Ознакомьтесь с командой nohup(1).

Запустите длительный процесс по nohup(1). Завершите сеанс работы. Снова войдите в систему и проверьте таблицу процессов. Поясните результат.

17. Определите uid процесса, каково минимальное значение и кому оно принадлежит. Каково минимальное и максимальное значение pid, каким процессам принадлежат? Проанализируйте множество системных процессов, как их отличить от прочих, перечислите назначение самых важных из них.

*Многонитевое функционирование*

18. Подготовьте программу, формирующую несколько нитей. Нити для эксперимента могут быть практически идентичны.

Например, каждая нить в цикле: выводит на печать собственное имя и инкрементирует переменную времени, после чего "засыпает" (sleep(5); sleep(1); -для первой и второй нитей соответственно), на экран (в файл) должно выводиться имя нити и количество пятисекундных (для первой) и секундных (для второй) интервалов функционирования каждой нити.

19. После запуска программы проанализируйте выполнение нитей, распределение во времени. Используйте для этого вывод таблицы процессов командой ps -axhf

Попробуйте удалить нить, зная ее идентификатор, командой kill.

Приведите и объясните результат.

20. Модифицируйте программу так, чтобы управление второй нитью осуществлялось посредством сигнала SIGUSR1 из первой нити.

На пятой секунде работы приложения удалите вторую нить. Для этого воспользуйтесь функцией pthread\_kill(t2, SIGUSR); (t2 - дескриптор второй нити).

В остальном программу можно не изменять. Проанализируйте полученные результаты.

21. Последняя модификация предполагает создание собственного обработчика сигнала, содержащего уведомление о начале его работы и возврат посредством функции pthread\_exit(NULL);

Сравните результаты, полученные после запуска этой модификации программы с результатами предыдущей.

22. Перехватите сигнал «CTRL C» для процесса и потока однократно, а также многократно с восстановлением исходного обработчика после нескольких раз срабатывания. Проделайте аналогичную работу для переназначения другой комбинации клавиш.

23. С помощью утилиты kill выведите список всех сигналов и дайте их краткую характеристику на основе документации ОС. Для чего предназначены сигналы с 32 по 64-й. Приведите пример их применения.

24. Проанализируйте процедуру планирования для процессов и потоков одного процесса.

24.1. Обоснуйте результат экспериментально.

24.2. Попробуйте процедуру планирования изменить. Подтвердите экспериментально, если изменение возможно.

24.3. Задайте нитям разные приоритеты программно и извне (объясните результат).

## **Выполнение работы.**

## **Информация о системе:**

Linux katya 6.5.0-28-generic #29~22.04.1-Ubuntu SMP PREEMPT\_DYNAMIC Thu Apr 4 14:39:20 UTC 2 x86\_64 x86\_64 x86\_64 GNU/Linux

**Задание 13.** *Взаимодействие родственных процессов*

13.1. Изменяя длительности выполнения процессов и параметры системных вызовов, рассмотрите 3 ситуации и получите соответствующие таблицы процессов:

а) процесс-отец запускает процесс-сын и ожидает его завершения;

б) процесс-отец запускает процесс-сын и, не ожидая его завершения, завершает свое выполнение. Зафиксируйте изменение родительского идентификатора процесса-сына;

в) процесс-отец запускает процесс-сын и не ожидает его завершения; процесс-сын завершает свое выполнение. Зафиксируйте появление процесса-зомби, для этого включите команду ps в программу father.c

13.2. Перенаправьте вывод не только на терминал, но и в файл. Организуйте программу многопроцессного функционирования так, чтобы результатом ее работы была демонстрация всех трех ситуаций с отображением в итоговом файле.

Реализована программа, которая сразу выполняет три заданные ситуации и выводит информацию в оба указанных источника.

(base) katya@katya:~/os/lb4/task13\_1$ cat **father.c**

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <wait.h>

#include <string.h>

int main(int argc, char \*argv[]){

int sid, pid, pid1, ppid, status;

char command[50];

if(argc < 2)

return -1;

pid = getpid();

ppid = getppid();

sid = getsid(pid);

//формирование команды для вывода информации о процессе в файл

sprintf(command, "ps xjf | grep \"STAT\\|%d\" > %s", sid, argv[1]);

printf("FATHER PARAMS: sid=%i, pid=%i, ppid=%i\n", sid, pid, ppid);

//создание дочерних процессов

if((pid1=fork()) == 0)

execl("son1", "son1", NULL);

if(fork() == 0)

execl("son2", "son2", argv[1], NULL);

if(fork() == 0)

execl("son3", "son3", NULL);

// выполнение команды записи в файл

system(command);

//ожидание завершения процесса с pid1 без блокировки

waitpid(pid1, &status, WNOHANG);

return 0;

}(base) katya@katya:~/os/lb4/task13\_1$ cat **son1.c**

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <wait.h>

#include <string.h>

void main(){

int pid, ppid;

pid = getpid();

ppid = getppid();

printf("SON\_1 PARAMS: pid=%i, ppid=%i\nFather creates and waits \n", pid, ppid);

sleep(3);

}(base) katya@katya:~/os/lb4/task13\_1$ cat **son2.c**

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#include <sched.h>

void main(int argc, char \*argv[]){

int pid, ppid;

pid = getpid();

ppid = getppid();

char command[50];

sprintf(command, "ps xjf | grep son2 >> %s", argv[1]);

printf("SON\_2 PARAMS: pid=%i, ppid=%i\nFather finished before son termination without waiting for it\n", pid, ppid);

sleep(20);

ppid = getppid();

printf("SON\_2 PARAMS ARE CHANGED: pid=%i, ppid=%i\n", pid, ppid);

system(command);

}(base) katya@katya:~/os/lb4/task13\_1$ cat **son3.c**

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <wait.h>

#include <string.h>

void main(){

int pid = getpid();

int ppid = getppid();

printf("SON\_3 PARAMS: pid=%i, ppid=%i\nson3 terminated - ZOMBIE\n", pid, ppid);

ppid = getppid();

printf("SON\_3 PARAMS: pid=%i, ppid=%i\n", pid, ppid);

(base) katya@katya:~/os/lb4/task13\_1$ gcc father.c -o father

(base) katya@katya:~/os/lb4/task13\_1$ gcc son1.c -o son1

(base) katya@katya:~/os/lb4/task13\_1$ gcc son2.c -o son2

(base) katya@katya:~/os/lb4/task13\_1$ gcc son3.c -o son3

(base) katya@katya:~/os/lb4/task13\_1$ .**/father res.txt**

FATHER PARAMS: sid=9071, pid=10547, ppid=9071

SON\_1 PARAMS: pid=10548, ppid=10547

Father creates and waits

SON\_2 PARAMS: pid=10549, ppid=10547

Father finished before son termination without waiting for it

SON\_3 PARAMS: pid=10550, ppid=10547

son3 terminated - ZOMBIE

SON\_3 PARAMS: pid=10550, ppid=10547

(base) katya@katya:~/os/lb4/task13\_1$ SON\_2 PARAMS ARE CHANGED: pid=10549, ppid=1506

(base) katya@katya:~/os/lb4/task13\_1$ **cat res.txt**

PPID PID PGID SID TTY TPGID STAT UID TIME COMMAND

9040 9071 9071 9071 pts/2 10547 Ss 1000 0:00 | \\_ bash

9071 10547 10547 9071 pts/2 10547 S+ 1000 0:00 | \\_ ./father res.txt

10547 10548 10547 9071 pts/2 10547 S+ 1000 0:00 | \\_ son1

10547 10549 10547 9071 pts/2 10547 S+ 1000 0:00 | \\_ son2 res.txt

10547 10550 10547 9071 pts/2 10547 Z+ 1000 0:00 | \\_ [son3] <defunct>

10547 10551 10547 9071 pts/2 10547 S+ 1000 0:00 | \\_ sh -c ps xjf | grep "STAT\|9071" > res.txt

10551 10552 10547 9071 pts/2 10547 R+ 1000 0:00 | \\_ ps xjf

10551 10553 10547 9071 pts/2 10547 S+ 1000 0:00 | \\_ grep STAT\|9071

1506 10549 10547 9071 pts/2 9071 S 1000 0:00 \\_ son2 res.txt

10549 10566 10547 9071 pts/2 9071 S 1000 0:00 \\_ sh -c ps xjf | grep son2 >> res.txt

10566 10568 10547 9071 pts/2 9071 S 1000 0:00 \\_ grep son2

Для того, чтобы потомок son2 существовал дольше, используется задержка в 20 секунд, и родитель завершается раньше. В результате этого потомок становится “самостоятельным” процессом с ppid=1056. Хотя в методическом пособии и сказано, что самостоятельность показывает ppid=1, следует учесть изменения в более современных дистрибутивах Linux.

Как видно из результатов, как только процесс-отец завершается, на консоли сразу появляется приглашение на ввод команды. А son2 продолжает свое выполнение в фоновом режиме. Т.к. Время выполнения son2 много дольше, то результат выполнения процесса-потомка, появляется уже после приглашения.

В файле res.txt можно отследить нормальное выполнение потомка son1, смена родителя son2 (ppid=1056) и его переход в самостоятельную ветку, состояние зомби для son3 (то есть процесс остается формально существующим, но ресурсы, отведенные для него, освобождаются).

*Управление процессами посредством сигналов*

**Задание 13.1** С помощью команды kill -l ознакомьтесь с перечнем сигналов, поддерживаемых процессами.

Ознакомьтесь с системными вызовами kill(2), signal(2).

Подготовьте программы следующего содержания:

а.) процесс father порождает процессы son1, son2, son3 и запускает на исполнение программные коды из соответствующих исполнительных файлов;

б.) далее родительский процесс осуществляет управление потомками, для этого он генерирует сигнал каждому пользовательскому процессу;

в.) в пользовательских процессах-потомках необходимо обеспечить: для son1 - реакцию на сигнал по умолчанию;

для son2 - реакцию игнорирования;

для son3 - перехватывание и обработку сигнала.

Сформируйте файл-проект из четырех файлов, откомпилируйте, запустите программу. Проанализируйте таблицу процессов до и после посылки сигналов с помощью системного вызова system("ps -s >> file"). Обратите внимание на реакцию, устанавливаемую для последнего потомка.

Системный вызов kill посылает сигналы указанным процессам. По умолчанию (если не указано имя или номер сигнала) посылается сигнал SIGTERM. Идентификатор процесса является аргументом для этой утилиты: если он больше нуля, то сигнал посылается процессу с указанным pid, если он равен нулю, то сигнал посылается всем процессам, принадлежащим пользователю, если он меньше нуля, то он воспринимается как идентификатор группы процессов, и тогда сигнал посылается всей группе.

Функция системного вызова signal заключается в том, чтобы задать определенные действия для программы в ответ на пришедший сигнал. В качестве действий можно задать следующие значения: SIG\_DFL, SIG\_IGN или указатель на собственную функцию обработки. SIG\_DFL означает, что процесс должен реагировать на сигнал, как задано по умолчанию (чаще всего это завершение процесса), SIG\_IGN означает, что нужно игнорировать сигнал.

(base) katya@katya:~/os/lb4/task13\_1$ kill -l

1) SIGHUP 2) SIGINT 3) SIGQUIT 4) SIGILL 5) SIGTRAP

6) SIGABRT 7) SIGBUS 8) SIGFPE 9) SIGKILL 10) SIGUSR1

11) SIGSEGV 12) SIGUSR2 13) SIGPIPE 14) SIGALRM 15) SIGTERM

16) SIGSTKFLT 17) SIGCHLD 18) SIGCONT 19) SIGSTOP 20) SIGTSTP

21) SIGTTIN 22) SIGTTOU 23) SIGURG 24) SIGXCPU 25) SIGXFSZ

26) SIGVTALRM 27) SIGPROF 28) SIGWINCH 29) SIGIO 30) SIGPWR

31) SIGSYS 34) SIGRTMIN 35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3

38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8

43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13

48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12

53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9 56) SIGRTMAX-8 57) SIGRTMAX-7

58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2

63) SIGRTMAX-1 64) SIGRTMAX

(base) katya@katya:~/os/lb4$ cd task13\_2\_1

(base) katya@katya:~/os/lb4/task13\_2\_1$ gcc signal.c -o signal

(base) katya@katya:~/os/lb4/task13\_2\_1$ gcc son1.c -o son1

(base) katya@katya:~/os/lb4/task13\_2\_1$ gcc son2.c -o son2

(base) katya@katya:~/os/lb4/task13\_2\_1$ gcc son3.c -o son3

(base) katya@katya:~/os/lb4/task13\_2\_1$ cat **signal.c**

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>

#include <sched.h>

int main(){

system(">res.txt");

int pid[3];

if ((pid[0] =fork()) == 0)

execl("son1", "son1", NULL);

if ((pid[1] = fork()) == 0)

execl("son2", "son2", NULL);

if ((pid[2] = fork()) == 0)

execl("son3", "son3", NULL);

system("ps -l >> res.txt");

system("echo \"\n\n\" >> res.txt");

kill(pid[0], SIGUSR1);

kill(pid[1], SIGUSR1);

kill(pid[2], SIGUSR1);

system("ps -l >> res.txt");

for (int i =0; i<3; i++)

wait(NULL);

return 0;

}

(base) katya@katya:~/os/lb4/task13\_2\_1$ cat **son1.c son2.c son3.c**

#include <signal.h>

#include <unistd.h>

//реакция на сигнал по умолчанию

int main(){

signal(SIGUSR1, SIG\_DFL);

sleep(5);

return 0;

}#include <signal.h>

#include <unistd.h>

//реакция игнорирования

int main(){

signal(SIGUSR1, SIG\_IGN);

sleep(5);

return 0;

}#include <signal.h>

#include <unistd.h>

void handler(int sig){}

//перехватываниее и обработка сигнала

int main(){

signal(SIGUSR1, handler);

sleep(5);

return 0;

}

В ходе программы детям отправляется сигнал SIGUSR1, реакцией по умолчанию на который является завершение процесса. Результат работы программы показывает, что у son1 изначально нет никакой маски, а после приема сигнала была изменена маска PENDING, процесс был завершен, значит сигнал обработался корректно. Маска IGNORED у son1 и маска CAUGHT у son2 была корректно установлена для нашего сигнала, который имеет 10 номер, то есть 10 бит в двоичной записи.

(base) katya@katya:~/os/lb4/task13\_2\_1$ cat res.txt

F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD

0 S 1000 32771 32740 0 80 0 - 2852 do\_wai pts/0 00:00:00 bash

0 S 1000 33292 32771 0 80 0 - 1514 do\_wai pts/0 00:00:00 signal

0 S 1000 33295 33292 0 80 0 - 1514 hrtime pts/0 00:00:00 son1

0 S 1000 33299 33292 0 80 0 - 1514 hrtime pts/0 00:00:00 son2

0 S 1000 33303 33292 0 80 0 - 1514 hrtime pts/0 00:00:00 son3

0 S 1000 33304 33292 0 80 0 - 1543 do\_wai pts/0 00:00:00 sh

4 R 1000 33305 33304 0 80 0 - 3998 - pts/0 00:00:00 ps

F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD

0 S 1000 32771 32740 0 80 0 - 2852 do\_wai pts/0 00:00:00 bash

0 S 1000 33292 32771 0 80 0 - 1514 do\_wai pts/0 00:00:00 signal

0 Z 1000 33295 33292 0 80 0 - 0 - pts/0 00:00:00 son1 <defunct>

0 S 1000 33299 33292 0 80 0 - 1514 hrtime pts/0 00:00:00 son2

0 Z 1000 33303 33292 0 80 0 - 0 - pts/0 00:00:00 son3 <defunct>

0 S 1000 33307 33292 0 80 0 - 1543 do\_wai pts/0 00:00:00 sh

4 R 1000 33308 33307 0 80 0 - 3998 - pts/0 00:00:00 ps

**Задание 13.2.** Организуйте посылку сигналов любым двум процессам, находящимся в разных состояниях: активном и пассивном, фиксируя моменты посылки и приема каждого сигнала с точностью до секунды. Приведите результаты в файле результатов.

(base) katya@katya:~/os/lb4/task13\_2\_2$ gcc father.c -o father

(base) katya@katya:~/os/lb4/task13\_2\_2$ gcc active\_son.c -o active\_son

(base) katya@katya:~/os/lb4/task13\_2\_2$ gcc passive\_son.c -o passive\_son

(base) katya@katya:~/os/lb4/task13\_2\_2$ cat **father.c**

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>

#include <sched.h>

#include <time.h>

#include <stdio.h>

int main(){

system(">res.txt");

//запуск детей

int pid[2];

if ((pid[0] =fork()) == 0)

execl("passive\_son", "passive\_son", NULL);

if ((pid[1] = fork()) == 0)

execl("active\_son", "active\_son", NULL);

//время для инициализации детей

sleep(1);

//проверяем в табл, что один процесс спит, а другой автивничает

system("ps -ft >> res.txt");

//отправляем сигнал и запоминаем это время

char temp\_str[100];

sprintf(temp\_str, "echo \"SEND PASSIVE %ld\n\" >> res.txt", time(NULL));

kill(pid[0], SIGUSR1);

//вывод времени отправки и отправляем сигнал

system(temp\_str);

sprintf(temp\_str, "echo \"SEND ACTIVE %ld\n\" >> res.txt", time(NULL));

kill(pid[1], SIGUSR1);

system(temp\_str);

for (int i =0; i<2; i++)

wait(NULL);

return 0;

}

(base) katya@katya:~/os/lb4/task13\_2\_2$ cat **active\_son.c**

#include <signal.h>

#include <unistd.h>

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

void handler(int signum){

//принимаем сигнал, запоминаем время, выводим в файл

char temp\_str[100];

sprintf(temp\_str, "echo \"GET ACTIVE %ld\n\" >> res.txt", time(NULL));

system(temp\_str);

exit(EXIT\_SUCCESS);

}

int main(){

//собственный разработчик

signal(SIGUSR1, handler);

for(int i =0; i < 10000000000; i++){}

return 0;

}(base) katya@katya:~/os/lb4/task13\_2\_2$ cat **passive\_son.c**

#include <signal.h>

#include <unistd.h>

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

void handler(int signum){

//принимаем сигнал, запоминаем время, выводим в файл

char temp\_str[100];

sprintf(temp\_str, "echo \"GET PASSIVE %ld\n\" >> res.txt", time(NULL));

system(temp\_str);

exit(EXIT\_SUCCESS);

}

int main(){

//собственный разработчик

signal(SIGUSR1, handler);

sleep(10);

return 0;

}

(base) katya@katya:~/os/lb4/task13\_2\_2$ ./father

(base) katya@katya:~/os/lb4/task13\_2\_2$ cat **res.txt**

PID TTY STAT TIME COMMAND

8349 pts/2 Ss 0:00 bash

8762 pts/2 S+ 0:00 \\_ ./father

8768 pts/2 S+ 0:00 \\_ passive\_son

8770 pts/2 R+ 0:01 \\_ active\_son

8771 pts/2 S+ 0:00 \\_ sh -c ps -ft >> res.txt

8772 pts/2 R+ 0:00 \\_ ps -ft

SEND PASSIVE 1716071917

GET PASSIVE 1716071917

SEND ACTIVE 1716071917

GET ACTIVE 1716071917

Реализованные программы помогают измерить время с точностью до секунд. Из полученного результата видно, что один процесс был S+, а другой R+. Можно было предположить, что спящему процессу нужно больше времени для обработки сигнала, однако видно, что получение и отправка сигнала происходило одновременно для обоих процессов.

**Задание 14.** Запустите в фоновом режиме несколько утилит, например: cat \*.c > myprog & lpr myprog & lpr intro&

Воспользуйтесь командой jobs для анализа списка заданий и очередности их выполнения. Позаботьтесь об уведомлении о завершении одного из заданий с помощью команды notify. Аргументом команды является номер задания. Верните невыполненные задания в приоритетный режим командой fg. Например: fg %. Отмените одно из невыполненных заданий.

(base) katya@katya:~$ sleep 60 & sleep 65 & sleep 70 &

[1] 9402

[2] 9403

[3] 9404

(base) katya@katya:~$ jobs -l

[1] 9402 Запущен sleep 60 &

[2]- 9403 Запущен sleep 65 &

[3]+ 9404 Запущен sleep 70 &

(base) katya@katya:~$ kill %1

(base) katya@katya:~$ jobs -l

[1] 9402 Завершено sleep 60

[2]- 9403 Запущен sleep 65 &

[3]+ 9404 Запущен sleep 70 &

(base) katya@katya:~$ jobs -l

[2]- 9403 Запущен sleep 65 &

[3]+ 9404 Запущен sleep 70 &

(base) katya@katya:~$ sleep 50 &

[4] 9411

(base) katya@katya:~$ jobs -l

[2] 9403 Запущен sleep 65 &

[3]- 9404 Запущен sleep 70 &

[4]+ 9411 Запущен sleep 50 &

(base) katya@katya:~$ fg %2

sleep 65

^Z

[2]+ Остановлен sleep 65

(base) katya@katya:~$ jobs -l

[2]+ 9403 Остановлено sleep 65

[3] 9404 Запущен sleep 70 &

[4]- 9411 Запущен sleep 50 &

(base) katya@katya:~$ bg %2

[2]+ sleep 65 &

(base) katya@katya:~$ jobs -l

[2] 9403 Запущен sleep 65 &

[3]- 9404 Запущен sleep 70 &

[4]+ 9411 Запущен sleep 50 &

(base) katya@katya:~$ jobs -l

[2] 9403 Запущен sleep 65 &

[3]- 9404 Запущен sleep 70 &

[4]+ 9411 Запущен sleep 50 &

(base) katya@katya:~$ jobs -l

[2] 9403 Завершён sleep 65

[3]- 9404 Завершён sleep 70

[4]+ 9411 Завершён sleep 50

(base) katya@katya:~$ jobs -l

(base) katya@katya:~$

Команда jobs в Unix/Linux используется для управления процессами, которые были запущены в фоновом режиме или остановлены. Она позволяет пользователю эффективно работать с несколькими процессами в одном терминальном окне.

Просмотр активных заданий: Просто введя jobs в терминале, можно увидеть список всех текущих заданий, включая их статус (запущено, остановлено) и идентификаторы заданий.

Перевод заданий в фоновый режим: Используя bg %n, где %n — это идентификатор задания, можно перевести задание в фоновый режим.

Приведение заданий к переднему плану: Команда fg %n принудительно переводит задание с идентификатором %n к переднему плану.

Остановка заданий: Для остановки задания можно использовать kill %n, где %n — это идентификатор задания.

**Задание 15.** Ознакомьтесь с выполнением команды и системного вызова nice(1) и getpriority(2).

Приведите примеры их использования в приложении. Определите границы приоритетов (создайте для этого программу). Есть ли разница в приоритетах для системных и пользовательских процессов, используются ли приоритеты реального времени? Каков пользовательский приоритет для запуска приложений из shell? Все ответы подкрепляйте экспериментально.

Системный вызов nice позволяет изменять базовый приоритет процесса. Приоритет процесса определяет, как быстро он будет получать время обработки CPU. Чем выше приоритет, тем больше вероятность того, что процесс будет выбран для выполнения.

Системный вызов getpriority используется для получения текущего приоритета процесса по определенным параметрам.

(base) katya@katya:~/os/lb4$ cat **15lim.c**

#include <stdio.h>

#include <sys/time.h>

#include <sys/resource.h>

int main(){

for(int i=-100; i<1; i++){

setpriority(PRIO\_PROCESS, 0, i);

int pr = getpriority(PRIO\_PROCESS, 0);

if (pr != i)

continue;

else{

printf("Нижняя граница = %d\n", pr);

break;

}

}

for(int i=1; i<100; i++){

setpriority(PRIO\_PROCESS, 0, i);

int pr = getpriority(PRIO\_PROCESS, 0);

if(pr == i)

continue;

else{

printf("Верхняя граница = %d\n", pr);

break;

}

}

return 0;

}(base) katya@katya:~/os/lb4$ gcc 15lim.c -o 15lim

(base) katya@katya:~/os/lb4$ ./15lim

Нижняя граница = 0

Верхняя граница = 19

(base) katya@katya:~/os/lb4$ sudo ./15lim

[sudo] пароль для katya:

Нижняя граница = -20

Верхняя граница = 19

Данная программа взята из методического пособия. Она определяет пределы nice: для суперпользователя это от -20 до 19, а для обычного пользователя это от 0 до 19. Таким образом можно сказать, что различия между приоритетами для системных и пользовательских процессов есть: системные процессы обычно имеют более высокие приоритеты, чем пользовательские, поскольку они отвечают за критически важные функции системы. Пользовательские приложения обычно запускаются с более низким приоритетом, чтобы не мешать работе системы.

(base) katya@katya:~/os/lb4$ cat **15ex.c**

#include <stdio.h>

#include <sys/time.h>

#include <sys/resource.h>

#include <unistd.h>

#include <stdlib.h>

int main(){

printf("Начальные приоритеты: %d\n", getpriority(PRIO\_PROCESS, 0));

system("ps -o pid,pri,ni,cmd,cls");

//делаем процесс менее приоритетным

nice(5);

printf("Измененные: %d\n", getpriority(PRIO\_PROCESS, 0));

system("ps -o pid,pri,ni,cmd,cls");

//попытка сделаеть более приоритетный процесс с отрицательным nice

// но это невозможно без прав суперпользователя

nice(-5);

printf("Измененные: %d\n", getpriority(PRIO\_PROCESS, 0));

system("ps -o pid,pri,ni,cmd,cls");

//снова меняем приоритет процесса

nice(9);

printf("Измененные: %d\n", getpriority(PRIO\_PROCESS, 0));

system("ps -o pid,pri,ni,cmd,cls");

return 0;

}

(base) katya@katya:~/os/lb4$ gcc 15ex.c -o 15ex

(base) katya@katya:~/os/lb4$ ./15ex

Начальные приоритеты: 0

PID PRI NI CMD CLS

9326 19 0 bash TS

11941 19 0 ./15ex TS

11942 19 0 sh -c ps -o pid,pri,ni,cmd, TS

11943 19 0 ps -o pid,pri,ni,cmd,cls TS

Измененные: 5

PID PRI NI CMD CLS

9326 19 0 bash TS

11941 14 5 ./15ex TS

11944 14 5 sh -c ps -o pid,pri,ni,cmd, TS

11945 14 5 ps -o pid,pri,ni,cmd,cls TS

Измененные: 5

PID PRI NI CMD CLS

9326 19 0 bash TS

11941 14 5 ./15ex TS

11946 14 5 sh -c ps -o pid,pri,ni,cmd, TS

11947 14 5 ps -o pid,pri,ni,cmd,cls TS

Измененные: 14

PID PRI NI CMD CLS

9326 19 0 bash TS

11941 5 14 ./15ex TS

11948 5 14 sh -c ps -o pid,pri,ni,cmd, TS

11949 5 14 ps -o pid,pri,ni,cmd,cls TS

(base) katya@katya:~/os/lb4$ sudo ./15ex

[sudo] пароль для katya:

Начальные приоритеты: 0

PID PRI NI CMD CLS

11951 19 0 sudo ./15ex TS

11952 19 0 ./15ex TS

11953 19 0 sh -c ps -o pid,pri,ni,cmd, TS

11954 19 0 ps -o pid,pri,ni,cmd,cls TS

Измененные: 5

PID PRI NI CMD CLS

11951 19 0 sudo ./15ex TS

11952 14 5 ./15ex TS

11955 14 5 sh -c ps -o pid,pri,ni,cmd, TS

11956 14 5 ps -o pid,pri,ni,cmd,cls TS

Измененные: 0

PID PRI NI CMD CLS

11951 19 0 sudo ./15ex TS

11952 19 0 ./15ex TS

11957 19 0 sh -c ps -o pid,pri,ni,cmd, TS

11958 19 0 ps -o pid,pri,ni,cmd,cls TS

Измененные: 9

PID PRI NI CMD CLS

11951 19 0 sudo ./15ex TS

11952 10 9 ./15ex TS

11959 10 9 sh -c ps -o pid,pri,ni,cmd, TS

11960 10 9 ps -o pid,pri,ni,cmd,cls TS

Данная программа создана для изменения приоритетов. Можно заметить, что результаты работы программы зависят от того, пользователь с каким правом ее запускает.

(base) katya@katya:~/os/lb4$ ps -o uid,pid,pri,ni,cmd,cls -xa

UID PID PRI NI CMD CLS

0 1 19 0 /sbin/init splash TS

0 2 19 0 [kthreadd] TS

0 3 39 -20 [rcu\_gp] TS

0 4 39 -20 [rcu\_par\_gp] TS

0 5 39 -20 [slub\_flushwq] TS

0 6 39 -20 [netns] TS

0 9 19 0 [kworker/0:1-events] TS

0 11 39 -20 [mm\_percpu\_wq] TS

0 12 19 0 [rcu\_tasks\_kthread] TS

0 13 19 0 [rcu\_tasks\_rude\_kthread] TS

0 14 19 0 [rcu\_tasks\_trace\_kthread] TS

0 15 19 0 [ksoftirqd/0] TS

0 16 19 0 [rcu\_preempt] TS

0 17 139 - [migration/0] FF

0 18 90 - [idle\_inject/0] FF

0 19 19 0 [cpuhp/0] TS

0 20 19 0 [cpuhp/2] TS

0 21 90 - [idle\_inject/2] FF

0 22 139 - [migration/2] FF

0 23 19 0 [ksoftirqd/2] TS

0 26 19 0 [cpuhp/4] TS

0 27 90 - [idle\_inject/4] FF

0 28 139 - [migration/4] FF

0 29 19 0 [ksoftirqd/4] TS

0 32 19 0 [cpuhp/6] TS

0 33 90 - [idle\_inject/6] FF

0 34 139 - [migration/6] FF

0 35 19 0 [ksoftirqd/6] TS

0 36 19 0 [kworker/6:0-mm\_percpu\_wq] TS

0 37 39 -20 [kworker/6:0H-kblockd] TS

0 38 19 0 [cpuhp/8] TS

0 39 90 - [idle\_inject/8] FF

0 40 139 - [migration/8] FF

0 41 19 0 [ksoftirqd/8] TS

0 44 19 0 [cpuhp/10] TS

0 45 90 - [idle\_inject/10] FF

0 46 139 - [migration/10] FF

0 47 19 0 [ksoftirqd/10] TS

0 50 19 0 [cpuhp/1] TS

0 51 90 - [idle\_inject/1] FF

0 52 139 - [migration/1] FF

0 53 19 0 [ksoftirqd/1] TS

0 56 19 0 [cpuhp/3] TS

0 57 90 - [idle\_inject/3] FF

0 58 139 - [migration/3] FF

0 59 19 0 [ksoftirqd/3] TS

0 62 19 0 [cpuhp/5] TS

0 63 90 - [idle\_inject/5] FF

0 64 139 - [migration/5] FF

0 65 19 0 [ksoftirqd/5] TS

0 68 19 0 [cpuhp/7] TS

0 69 90 - [idle\_inject/7] FF

0 70 139 - [migration/7] FF

0 71 19 0 [ksoftirqd/7] TS

0 73 39 -20 [kworker/7:0H-kblockd] TS

0 74 19 0 [cpuhp/9] TS

0 75 90 - [idle\_inject/9] FF

0 76 139 - [migration/9] FF

0 77 19 0 [ksoftirqd/9] TS

0 80 19 0 [cpuhp/11] TS

0 81 90 - [idle\_inject/11] FF

0 82 139 - [migration/11] FF

0 83 19 0 [ksoftirqd/11] TS

0 86 19 0 [kdevtmpfs] TS

0 87 39 -20 [inet\_frag\_wq] TS

0 89 19 0 [kauditd] TS

0 90 19 0 [khungtaskd] TS

0 91 19 0 [oom\_reaper] TS

0 93 39 -20 [writeback] TS

0 94 19 0 [kcompactd0] TS

0 95 14 5 [ksmd] TS

0 96 0 19 [khugepaged] TS

0 97 39 -20 [kintegrityd] TS

0 98 39 -20 [kblockd] TS

0 99 39 -20 [blkcg\_punt\_bio] TS

0 100 19 0 [kworker/1:1-pm] TS

0 101 39 -20 [tpm\_dev\_wq] TS

0 102 39 -20 [ata\_sff] TS

0 103 39 -20 [md] TS

0 104 39 -20 [md\_bitmap] TS

0 105 39 -20 [edac-poller] TS

0 106 39 -20 [devfreq\_wq] TS

0 107 90 - [watchdogd] FF

0 109 39 -20 [kworker/0:1H-ttm] TS

0 110 90 - [irq/25-AMD-Vi] FF

0 111 19 0 [kswapd0] TS

0 112 19 0 [ecryptfs-kthread] TS

0 113 39 -20 [kthrotld] TS

0 116 19 0 [kworker/4:1-events] TS

0 117 19 0 [kworker/5:1-mm\_percpu\_wq] TS

0 120 19 0 [kworker/9:1-mm\_percpu\_wq] TS

0 123 39 -20 [acpi\_thermal\_pm] TS

0 124 39 -20 [mld] TS

0 125 39 -20 [ipv6\_addrconf] TS

0 133 39 -20 [kstrp] TS

0 135 39 -20 [kworker/u33:0-hci0] TS

0 139 39 -20 [charger\_manager] TS

0 140 90 - [irq/26-ACPI:Event] FF

0 141 90 - [irq/27-ACPI:Event] FF

0 142 90 - [irq/28-ACPI:Event] FF

0 143 90 - [irq/29-ACPI:Event] FF

0 144 90 - [irq/30-ACPI:Event] FF

0 145 90 - [irq/31-ACPI:Event] FF

0 146 90 - [irq/32-ACPI:Event] FF

0 147 90 - [irq/33-ACPI:Event] FF

0 148 90 - [irq/34-ACPI:Event] FF

0 170 39 -20 [kworker/7:1H-ttm] TS

0 186 39 -20 [kworker/8:1H-kblockd] TS

0 189 39 -20 [kworker/10:1H-kblockd] TS

0 203 90 - [irq/38-ASUE1211:00] FF

0 226 39 -20 [nvme-wq] TS

0 227 39 -20 [nvme-reset-wq] TS

0 228 39 -20 [nvme-delete-wq] TS

0 229 39 -20 [nvme-auth-wq] TS

0 255 19 0 [jbd2/nvme0n1p6-8] TS

0 256 39 -20 [ext4-rsv-conver] TS

0 295 20 -1 /lib/systemd/systemd-journa TS

0 366 19 0 /lib/systemd/systemd-udevd TS

0 414 39 -20 [cryptd] TS

0 423 39 -20 [led\_workqueue] TS

0 534 39 -20 [cfg80211] TS

0 543 39 -20 [kworker/u33:1-hci0] TS

0 567 39 -20 [amd\_iommu\_v2] TS

0 580 19 0 [kworker/10:2-events] TS

0 581 19 0 [napi/phy0-8193] TS

0 582 19 0 [napi/phy0-8194] TS

0 583 19 0 [napi/phy0-8195] TS

0 585 19 0 [jbd2/nvme0n1p7-8] TS

0 586 39 -20 [ext4-rsv-conver] TS

108 614 19 0 /lib/systemd/systemd-oomd TS

101 617 19 0 /lib/systemd/systemd-resolv TS

103 618 19 0 /lib/systemd/systemd-timesy TS

0 670 41 - [mt76-tx phy0] FF

0 672 39 -20 [amdgpu-reset-de] TS

0 674 19 0 /usr/libexec/accounts-daemo TS

0 675 19 0 /usr/sbin/acpid TS

114 677 19 0 avahi-daemon: running [katy TS

0 678 19 0 /usr/lib/bluetooth/bluetoot TS

0 679 19 0 /usr/sbin/cron -f -P TS

102 680 19 0 @dbus-daemon --system --add TS

0 681 19 0 /usr/sbin/NetworkManager -- TS

0 689 19 0 /usr/sbin/irqbalance --fore TS

0 691 19 0 /usr/bin/python3 /usr/bin/n TS

0 692 19 0 /usr/libexec/polkitd --no-d TS

0 693 19 0 /usr/libexec/power-profiles TS

104 694 19 0 /usr/sbin/rsyslogd -n -iNON TS

0 696 19 0 /usr/lib/snapd/snapd TS

0 697 19 0 /usr/libexec/switcheroo-con TS

0 698 19 0 /lib/systemd/systemd-logind TS

0 700 19 0 /usr/libexec/udisks2/udisks TS

0 701 19 0 /sbin/wpa\_supplicant -u -s TS

114 703 19 0 avahi-daemon: chroot helper TS

0 729 39 -20 [ttm] TS

0 770 39 -20 [amdgpu\_dm\_hpd\_r] TS

0 771 39 -20 [amdgpu\_dm\_hpd\_r] TS

0 772 39 -20 [dm\_vblank\_contr] TS

0 773 19 0 /usr/sbin/ModemManager TS

0 789 19 0 /usr/sbin/cupsd -l TS

0 792 19 0 /usr/bin/python3 /usr/share TS

0 799 90 - [card0-crtc0] FF

0 800 90 - [card0-crtc1] FF

0 801 90 - [card0-crtc2] FF

0 802 90 - [card0-crtc3] FF

0 803 41 - [gfx\_low] FF

0 804 41 - [gfx\_high] FF

0 805 41 - [comp\_1.0.0] FF

0 806 41 - [comp\_1.1.0] FF

0 807 41 - [comp\_1.2.0] FF

0 808 41 - [comp\_1.3.0] FF

0 809 41 - [comp\_1.0.1] FF

0 810 41 - [comp\_1.1.1] FF

0 811 41 - [comp\_1.2.1] FF

0 812 41 - [comp\_1.3.1] FF

0 813 41 - [sdma0] FF

0 814 41 - [vcn\_dec] FF

0 815 41 - [vcn\_enc0] FF

0 816 41 - [vcn\_enc1] FF

0 817 41 - [jpeg\_dec] FF

0 898 19 0 /usr/sbin/gdm3 TS

116 961 18 1 /usr/libexec/rtkit-daemon TS

0 1100 19 0 /usr/sbin/cups-browsed TS

113 1103 19 0 /usr/sbin/kerneloops --test TS

113 1105 19 0 /usr/sbin/kerneloops TS

0 1199 19 0 /usr/libexec/upowerd TS

0 1236 39 -20 [kworker/6:2H-ttm] TS

0 1277 19 0 /usr/libexec/packagekitd TS

123 1409 19 0 /usr/libexec/colord TS

0 1486 19 0 [kworker/11:3-events] TS

0 1493 19 0 gdm-session-worker [pam/gdm TS

1000 1509 19 0 /lib/systemd/systemd --user TS

1000 1510 19 0 (sd-pam) TS

1000 1516 30 -11 /usr/bin/pipewire TS

1000 1517 19 0 /usr/bin/pipewire-media-ses TS

1000 1518 30 -11 /usr/bin/pulseaudio --daemo TS

1000 1529 19 0 /usr/bin/dbus-daemon --sess TS

1000 1536 19 0 /usr/libexec/gvfsd TS

1000 1544 19 0 /usr/libexec/gvfsd-fuse /ru TS

1000 1559 19 0 /usr/libexec/xdg-document-p TS

1000 1564 19 0 /usr/libexec/xdg-permission TS

1000 1565 19 0 /usr/bin/gnome-keyring-daem TS

0 1573 19 0 fusermount3 -o rw,nosuid,no TS

1000 1584 0 - /usr/libexec/tracker-miner- IDL

0 1599 29 -10 [krfcommd] TS

1000 1600 19 0 /usr/libexec/gvfs-udisks2-v TS

1000 1605 19 0 /usr/libexec/gvfs-afc-volum TS

1000 1610 19 0 /usr/libexec/gvfs-goa-volum TS

1000 1614 19 0 /usr/libexec/goa-daemon TS

1000 1621 19 0 /usr/libexec/gdm-x-session TS

1000 1623 19 0 /usr/libexec/goa-identity-s TS

1000 1625 19 0 /usr/lib/xorg/Xorg vt2 -dis TS

1000 1627 19 0 /usr/libexec/gvfs-gphoto2-v TS

1000 1637 19 0 /usr/libexec/gvfs-mtp-volum TS

1000 1691 19 0 /usr/libexec/gnome-session- TS

1000 1770 19 0 /usr/libexec/at-spi-bus-lau TS

1000 1776 19 0 /usr/bin/dbus-daemon --conf TS

1000 1789 19 0 /usr/libexec/gnome-session- TS

1000 1801 19 0 /usr/libexec/gnome-session- TS

1000 1824 19 0 /usr/bin/gnome-shell TS

1000 1849 19 0 /usr/libexec/gnome-shell-ca TS

1000 1853 19 0 /usr/libexec/dconf-service TS

1000 1861 19 0 /usr/libexec/evolution-sour TS

1000 1862 19 0 /snap/snapd-desktop-integra TS

1000 1920 19 0 /snap/snapd-desktop-integra TS

1000 1924 19 0 /usr/libexec/gvfsd-trash -- TS

1000 1931 19 0 /usr/libexec/evolution-cale TS

1000 1941 19 0 /usr/bin/gjs /usr/share/gno TS

1000 1943 19 0 /usr/libexec/at-spi2-regist TS

1000 1967 19 0 /usr/libexec/evolution-addr TS

1000 1970 19 0 sh -c /usr/bin/ibus-daemon TS

1000 1972 19 0 /usr/libexec/gsd-a11y-setti TS

1000 1973 19 0 /usr/bin/ibus-daemon --pane TS

1000 1975 19 0 /usr/libexec/gsd-color TS

1000 1977 19 0 /usr/libexec/gsd-datetime TS

1000 1981 19 0 /usr/libexec/gsd-housekeepi TS

1000 1984 19 0 /usr/libexec/gsd-keyboard TS

1000 1986 19 0 /usr/libexec/gsd-media-keys TS

1000 1988 19 0 /usr/libexec/gsd-power TS

1000 1992 19 0 /usr/libexec/gsd-print-noti TS

1000 1995 19 0 /usr/libexec/gsd-rfkill TS

1000 1997 19 0 /usr/libexec/gsd-screensave TS

1000 2003 19 0 /usr/libexec/gsd-sharing TS

1000 2004 19 0 /usr/libexec/gsd-smartcard TS

1000 2007 19 0 /usr/libexec/gsd-sound TS

1000 2012 19 0 /usr/libexec/gsd-wacom TS

1000 2015 19 0 /usr/libexec/gsd-xsettings TS

1000 2031 19 0 /usr/libexec/evolution-data TS

1000 2034 19 0 /usr/libexec/gsd-disk-utili TS

1000 2038 19 0 /usr/libexec/ibus-dconf TS

1000 2047 19 0 /usr/libexec/ibus-extension TS

1000 2075 19 0 /usr/libexec/gsd-printer TS

1000 2108 19 0 /usr/libexec/ibus-x11 --kil TS

1000 2114 19 0 /usr/libexec/ibus-portal TS

1000 2121 19 0 /usr/bin/gjs /usr/share/gno TS

1000 2147 19 0 /usr/libexec/ibus-engine-si TS

1000 2156 19 0 /usr/libexec/xdg-desktop-po TS

1000 2173 19 0 /snap/snap-store/1113/usr/b TS

1000 2184 19 0 /usr/libexec/xdg-desktop-po TS

1000 2342 19 0 /usr/libexec/xdg-desktop-po TS

0 2368 39 -20 [kworker/11:3H-ttm] TS

0 2423 19 0 /usr/libexec/fwupd/fwupd TS

0 2428 39 -20 [kworker/10:2H-ttm] TS

1000 2434 19 0 /usr/libexec/gvfsd-metadata TS

1000 2607 19 0 update-notifier TS

1000 2826 19 0 /opt/google/chrome/chrome TS

1000 2831 19 0 cat TS

1000 2832 19 0 cat TS

1000 2834 19 0 /opt/google/chrome/chrome\_c TS

1000 2836 19 0 /opt/google/chrome/chrome\_c TS

1000 2842 19 0 /opt/google/chrome/chrome - TS

1000 2843 19 0 /opt/google/chrome/chrome - TS

1000 2844 19 0 /snap/telegram-desktop/5820 TS

1000 2856 19 0 /opt/google/chrome/nacl\_hel TS

1000 2863 19 0 /opt/google/chrome/chrome - TS

1000 2965 19 0 /opt/google/chrome/chrome - TS

1000 2970 19 0 /opt/google/chrome/chrome - TS

1000 2980 19 0 /opt/google/chrome/chrome - TS

1000 3100 19 0 /opt/google/chrome/chrome - TS

1000 3111 19 0 /opt/google/chrome/chrome - TS

1000 3245 19 0 /opt/google/chrome/chrome - TS

1000 3966 19 0 /opt/google/chrome/chrome - TS

1000 4200 19 0 /opt/google/chrome/chrome - TS

1000 4216 19 0 /opt/google/chrome/chrome - TS

1000 4425 19 0 /usr/bin/nautilus --gapplic TS

1000 4573 19 0 /opt/google/chrome/chrome - TS

1000 4637 19 0 evince /home/katya/Загрузки TS

1000 4650 19 0 /usr/libexec/evinced TS

1000 4688 19 0 /usr/lib/libreoffice/progra TS

1000 4704 19 0 /usr/lib/libreoffice/progra TS

1000 5456 19 0 /snap/code/159/usr/share/co TS

1000 5458 19 0 /snap/code/159/usr/share/co TS

1000 5459 19 0 /snap/code/159/usr/share/co TS

1000 5472 19 0 /snap/code/159/usr/share/co TS

1000 5498 19 0 /snap/code/159/usr/share/co TS

1000 5520 19 0 /snap/code/159/usr/share/co TS

1000 5552 19 0 /snap/code/159/usr/share/co TS

1000 5593 19 0 /snap/code/159/usr/share/co TS

1000 5602 19 0 /snap/code/159/usr/share/co TS

1000 5603 19 0 /snap/code/159/usr/share/co TS

1000 5639 19 0 /snap/code/159/usr/share/co TS

1000 5680 19 0 /home/katya/.vscode/extensi TS

1000 5762 19 0 /usr/bin/bash --init-file / TS

1000 5896 19 0 /usr/bin/gedit --gapplicati TS

0 6513 19 0 [kworker/7:2-events] TS

0 6901 19 0 [kworker/8:0-events] TS

0 7309 19 0 [kworker/3:2-events] TS

0 7698 19 0 [kworker/9:0] TS

1000 8189 19 0 gjs /usr/share/gnome-shell/ TS

0 8299 39 -20 [kworker/3:0H-kblockd] TS

0 8646 39 -20 [kworker/11:0H-ttm] TS

0 8719 39 -20 [kworker/5:2H-ttm] TS

0 8791 39 -20 [kworker/8:0H-ttm] TS

0 8827 39 -20 [kworker/5:3H-kblockd] TS

0 8885 39 -20 [kworker/3:1H-ttm] TS

1000 8915 19 0 /opt/google/chrome/chrome - TS

0 9104 19 0 [kworker/u32:0-events\_power TS

0 9151 19 0 [kworker/2:1-mm\_percpu\_wq] TS

1000 9295 19 0 /usr/libexec/gnome-terminal TS

1000 9326 19 0 bash TS

0 9419 19 0 [kworker/u32:1-events\_power TS

0 9510 39 -20 [kworker/1:1H-kblockd] TS

0 9512 39 -20 [kworker/1:4H-ttm] TS

0 9549 39 -20 [kworker/2:2H-ttm] TS

0 9551 39 -20 [kworker/2:4H-ttm] TS

0 9553 39 -20 [kworker/0:2H-ttm] TS

0 9555 39 -20 [kworker/4:2H-ttm] TS

1000 9612 19 0 /opt/google/chrome/chrome - TS

0 9700 19 0 [kworker/5:0] TS

1000 9720 19 0 /opt/google/chrome/chrome - TS

1000 9825 19 0 /home/katya/.vscode/extensi TS

0 9870 19 0 [kworker/0:0-mm\_percpu\_wq] TS

0 10015 19 0 [kworker/6:1-events] TS

0 10286 39 -20 [kworker/9:2H-kblockd] TS

0 10287 39 -20 [kworker/9:3H-ttm] TS

0 10321 19 0 [kworker/10:0-events] TS

0 10349 19 0 [kworker/7:1-events] TS

0 10420 19 0 [kworker/u32:2-events\_unbou TS

0 10424 19 0 [kworker/1:0] TS

1000 10437 19 0 /home/katya/.vscode/extensi TS

0 10582 19 0 [kworker/4:2-events] TS

0 10786 19 0 [kworker/11:0] TS

0 10851 19 0 [kworker/2:0-events] TS

0 10906 19 0 [kworker/3:1] TS

0 11413 19 0 [kworker/u32:3+events\_unbou TS

0 11690 19 0 [kworker/8:1-events] TS

0 11837 39 -20 [kworker/4:0H-kblockd] TS

0 11961 19 0 [kworker/u32:4] TS

0 11962 39 -20 [kworker/9:0H] TS

0 11975 39 -20 [kworker/11:1H-ttm] TS

0 11976 39 -20 [kworker/3:2H] TS

0 11977 39 -20 [kworker/2:0H] TS

0 11978 39 -20 [kworker/2:1H-ttm] TS

1000 11993 19 0 /opt/google/chrome/chrome - TS

1000 12010 19 0 ps -o uid,pid,pri,ni,cmd,cl TS

Представлены все процессы с их приоритетами. Видно, что у большинства процессов столбец PRI с 19, но у процессов, запущенных от лица root, часто встречается -20 в столбце NI, что говорит о повышенном приоритете для данных процессов.

Приоритеты реального времени (RT) используются для процессов, которым требуется гарантированный доступ к ресурсам. Они отличаются от обычных приоритетов и обеспечивают более строгую гарантию выполнения. Нулевое значение в столбце PRI говорит о приоритете реального времени. Таких процессов мало, но все же они есть. Например, у меня их всего 2.

Из shell все пользовательские процессы запускаются по умолчанию с приоритетом 19 и nice 0.

**Задание 16.** Ознакомьтесь с командой nohup(1).

Запустите длительный процесс по nohup(1). Завершите сеанс работы. Снова войдите в систему и проверьте таблицу процессов. Поясните результат.

Команда nohup позволяет запускать процессы таким образом, что они продолжают работать даже после выхода из сессии. Это достигается за счет игнорирования сигнала SIGHUP, который обычно приводит к завершению процессов при закрытии терминала. Когда используется nohup, вывод процесса перенаправляется в файл nohup.out в текущем каталоге, если не указано иное.

(base) katya@katya:~$ cd os/lb4

(base) katya@katya:~/os/lb4$ cat task16.c

#include <stdio.h>

#include <unistd.h>

int main(){

sleep(50);

printf("Nohup\n");

return 0;

}

(base) katya@katya:~/os/lb4$ gcc task16.c -o task16

Используем утилиту, процесс виден в списке:

(base) katya@katya:~/os/lb4$ nohup ./task16 &

[1] 13774

(base) katya@katya:~/os/lb4$ nohup: ввод игнорируется, вывод добавляется в 'nohup.out'

ps

PID TTY TIME CMD

13717 pts/2 00:00:00 bash

13774 pts/2 00:00:00 task16

13775 pts/2 00:00:00 ps

После закрытия терминала процесс тоже виден как активный:

(base) katya@katya:~$ ps aux | grep task16

katya 13774 0.0 0.0 6056 2432 ? S 03:52 0:00 ./task16

katya 13893 0.0 0.0 12596 3840 pts/2 S+ 03:53 0:00 grep --color=auto task16

Запускаем без использования утилиты:

(base) katya@katya:~/os/lb4$ ./task16 &

[1] 13913

(base) katya@katya:~/os/lb4$ ps aux | grep task16

katya 13913 0.0 0.0 6056 2432 pts/2 S 03:54 0:00 ./task16

katya 13915 0.0 0.0 12596 3840 pts/2 S+ 03:54 0:00 grep --color=auto task16

(base) katya@katya:~/os/lb4$ cat nohup.out

Nohup

После закрытия терминала запущенный процесс уже не видно. Таким образом практика сошлась с теорией: Процесс, запущенный с помощью nohup, продолжит работу даже после выхода из сессии, благодаря тому, что он не получает сигнал SIGHUP, который обычно приводит к его завершению. Вывод процесса будет сохранен в файле nohup.out, если не было указано другое место для записи вывода. Это позволяет продолжать работу длительных процессов без необходимости оставаться в сессии или перезапускать их при каждом входе в систему.

**Задание 17.** Определите uid процесса, каково минимальное значение и кому оно принадлежит. Каково минимальное и максимальное значение pid, каким процессам принадлежат? Проанализируйте множество системных процессов, как их отличить от прочих, перечислите назначение самых важных из них.

UID (User Identifier) процесса определяет владельца процесса. Минимальное значение UID равно 0 и принадлежит суперпользователю (root), который имеет полный контроль над системой. Все остальные пользователи имеют UID, начинающиеся с 1 и увеличивающиеся по мере добавления новых пользователей в систему. Все UID можно посмотреть в /etc/passwd в 3 “столбце”. UID реальных пользователей у меня начинаются с 1000.

(base) katya@katya:~/os/lb4$ cat /etc/passwd

root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin

bin:x:2:2:bin:/bin:/usr/sbin/nologin

sys:x:3:3:sys:/dev:/usr/sbin/nologin

sync:x:4:65534:sync:/bin:/bin/sync

games:x:5:60:games:/usr/games:/usr/sbin/nologin

man:x:6:12:man:/var/cache/man:/usr/sbin/nologin

lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin

mail:x:8:8:mail:/var/mail:/usr/sbin/nologin

news:x:9:9:news:/var/spool/news:/usr/sbin/nologin

uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin

proxy:x:13:13:proxy:/bin:/usr/sbin/nologin

www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin

backup:x:34:34:backup:/var/backups:/usr/sbin/nologin

list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin

irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin

gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin

nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin

systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin

systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin

messagebus:x:102:105::/nonexistent:/usr/sbin/nologin

systemd-timesync:x:103:106:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin

syslog:x:104:111::/home/syslog:/usr/sbin/nologin

\_apt:x:105:65534::/nonexistent:/usr/sbin/nologin

tss:x:106:113:TPM software stack,,,:/var/lib/tpm:/bin/false

uuidd:x:107:116::/run/uuidd:/usr/sbin/nologin

systemd-oom:x:108:117:systemd Userspace OOM Killer,,,:/run/systemd:/usr/sbin/nologin

tcpdump:x:109:118::/nonexistent:/usr/sbin/nologin

avahi-autoipd:x:110:119:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin

usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin

dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin

kernoops:x:113:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin

avahi:x:114:121:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin

cups-pk-helper:x:115:122:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin

rtkit:x:116:123:RealtimeKit,,,:/proc:/usr/sbin/nologin

whoopsie:x:117:124::/nonexistent:/bin/false

sssd:x:118:125:SSSD system user,,,:/var/lib/sss:/usr/sbin/nologin

speech-dispatcher:x:119:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false

fwupd-refresh:x:120:126:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin

nm-openvpn:x:121:127:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin

saned:x:122:129::/var/lib/saned:/usr/sbin/nologin

colord:x:123:130:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin

geoclue:x:124:131::/var/lib/geoclue:/usr/sbin/nologin

pulse:x:125:132:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin

gnome-initial-setup:x:126:65534::/run/gnome-initial-setup/:/bin/false

hplip:x:127:7:HPLIP system user,,,:/run/hplip:/bin/false

gdm:x:128:134:Gnome Display Manager:/var/lib/gdm3:/bin/false

katya:x:1000:1000:Katya,,,:/home/katya:/bin/bash

kotik:x:1001:1001::/home/kotik:/bin/bash

Минимальное значение PID: На большинстве современных систем Linux минимальное значение PID равно 1 и обычно принадлежит процессу init, который является родительским процессом для всех других процессов в системе.

Максимальное значение PID: Максимальное значение PID зависит от конфигурации системы и может быть найдено, посмотрев содержимое файла /proc/sys/kernel/pid\_max. Этот файл содержит максимальное значение PID, которое система может создать.

(base) katya@katya:~/os/lb4$ cat /proc/sys/kernel/pid\_max

4194304

Системные процессы обычно имеют низкие PID и принадлежат root (UID=0). Они отвечают за различные аспекты работы операционной системы, такие как управление памятью, сетью, дисковым устройством и т.д.

init (PID=1): Инициализирует систему и управляет всеми другими процессами.

kthreadd (PID=2): Основной поток ядра, отвечает за многие внутренние функции ядра.

ksoftirqd (PID=15…): запускается, когда требуется уменьшить нагрузку на IRQ (Interrupt Request)

(base) katya@katya:~/os/lb4$ ps -o uid,pid,pri,ni,cmd,cls,f -e

UID PID PRI NI CMD CLS F

0 1 19 0 /sbin/init splash TS 4

0 2 19 0 [kthreadd] TS 1

0 3 39 -20 [rcu\_gp] TS 1

0 4 39 -20 [rcu\_par\_gp] TS 1

0 5 39 -20 [slub\_flushwq] TS 1

0 6 39 -20 [netns] TS 1

0 11 39 -20 [mm\_percpu\_wq] TS 1

0 12 19 0 [rcu\_tasks\_kthread] TS 1

0 13 19 0 [rcu\_tasks\_rude\_kthread] TS 1

0 14 19 0 [rcu\_tasks\_trace\_kthread] TS 1

0 15 19 0 [ksoftirqd/0] TS 1

0 16 19 0 [rcu\_preempt] TS 1

0 17 139 - [migration/0] FF 1

0 18 90 - [idle\_inject/0] FF 1

0 19 19 0 [cpuhp/0] TS 1

0 20 19 0 [cpuhp/2] TS 5

0 21 90 - [idle\_inject/2] FF 1

0 22 139 - [migration/2] FF 1

0 23 19 0 [ksoftirqd/2] TS 1

0 26 19 0 [cpuhp/4] TS 5

0 27 90 - [idle\_inject/4] FF 1

0 28 139 - [migration/4] FF 1

0 29 19 0 [ksoftirqd/4] TS 1

0 32 19 0 [cpuhp/6] TS 5

0 33 90 - [idle\_inject/6] FF 1

0 34 139 - [migration/6] FF 1

0 35 19 0 [ksoftirqd/6] TS 1

0 36 19 0 [kworker/6:0-events] TS 1

0 37 39 -20 [kworker/6:0H-ttm] TS 1

0 38 19 0 [cpuhp/8] TS 5

0 39 90 - [idle\_inject/8] FF 1

0 40 139 - [migration/8] FF 1

0 41 19 0 [ksoftirqd/8] TS 1

0 44 19 0 [cpuhp/10] TS 5

0 45 90 - [idle\_inject/10] FF 1

0 46 139 - [migration/10] FF 1

0 47 19 0 [ksoftirqd/10] TS 1

0 50 19 0 [cpuhp/1] TS 5

0 51 90 - [idle\_inject/1] FF 1

0 52 139 - [migration/1] FF 1

0 53 19 0 [ksoftirqd/1] TS 1

0 56 19 0 [cpuhp/3] TS 5

0 57 90 - [idle\_inject/3] FF 1

0 58 139 - [migration/3] FF 1

0 59 19 0 [ksoftirqd/3] TS 1

0 62 19 0 [cpuhp/5] TS 5

0 63 90 - [idle\_inject/5] FF 1

0 64 139 - [migration/5] FF 1

0 65 19 0 [ksoftirqd/5] TS 1

0 68 19 0 [cpuhp/7] TS 5

0 69 90 - [idle\_inject/7] FF 1

0 70 139 - [migration/7] FF 1

0 71 19 0 [ksoftirqd/7] TS 1

0 73 39 -20 [kworker/7:0H-ttm] TS 1

0 74 19 0 [cpuhp/9] TS 5

0 75 90 - [idle\_inject/9] FF 1

0 76 139 - [migration/9] FF 1

0 77 19 0 [ksoftirqd/9] TS 1

0 80 19 0 [cpuhp/11] TS 5

0 81 90 - [idle\_inject/11] FF 1

0 82 139 - [migration/11] FF 1

0 83 19 0 [ksoftirqd/11] TS 1

0 86 19 0 [kdevtmpfs] TS 5

0 87 39 -20 [inet\_frag\_wq] TS 1

0 89 19 0 [kauditd] TS 1

0 90 19 0 [khungtaskd] TS 1

0 91 19 0 [oom\_reaper] TS 1

0 93 39 -20 [writeback] TS 1

0 94 19 0 [kcompactd0] TS 1

0 95 14 5 [ksmd] TS 1

0 96 0 19 [khugepaged] TS 1

0 97 39 -20 [kintegrityd] TS 1

0 98 39 -20 [kblockd] TS 1

0 99 39 -20 [blkcg\_punt\_bio] TS 1

0 100 19 0 [kworker/1:1-mm\_percpu\_wq] TS 1

0 101 39 -20 [tpm\_dev\_wq] TS 1

0 102 39 -20 [ata\_sff] TS 1

0 103 39 -20 [md] TS 1

0 104 39 -20 [md\_bitmap] TS 1

0 105 39 -20 [edac-poller] TS 1

0 106 39 -20 [devfreq\_wq] TS 1

0 107 90 - [watchdogd] FF 1

0 110 90 - [irq/25-AMD-Vi] FF 1

0 111 19 0 [kswapd0] TS 1

0 112 19 0 [ecryptfs-kthread] TS 1

0 113 39 -20 [kthrotld] TS 1

0 117 19 0 [kworker/5:1-mm\_percpu\_wq] TS 1

0 120 19 0 [kworker/9:1-mm\_percpu\_wq] TS 1

0 123 39 -20 [acpi\_thermal\_pm] TS 1

0 124 39 -20 [mld] TS 1

0 125 39 -20 [ipv6\_addrconf] TS 1

0 133 39 -20 [kstrp] TS 1

0 135 39 -20 [kworker/u33:0-hci0] TS 1

0 139 39 -20 [charger\_manager] TS 1

0 140 90 - [irq/26-ACPI:Event] FF 1

0 141 90 - [irq/27-ACPI:Event] FF 1

0 142 90 - [irq/28-ACPI:Event] FF 1

0 143 90 - [irq/29-ACPI:Event] FF 1

0 144 90 - [irq/30-ACPI:Event] FF 1

0 145 90 - [irq/31-ACPI:Event] FF 1

0 146 90 - [irq/32-ACPI:Event] FF 1

0 147 90 - [irq/33-ACPI:Event] FF 1

0 148 90 - [irq/34-ACPI:Event] FF 1

0 170 39 -20 [kworker/7:1H-kblockd] TS 1

0 189 39 -20 [kworker/10:1H-kblockd] TS 1

0 203 90 - [irq/38-ASUE1211:00] FF 1

0 226 39 -20 [nvme-wq] TS 1

0 227 39 -20 [nvme-reset-wq] TS 1

0 228 39 -20 [nvme-delete-wq] TS 1

0 229 39 -20 [nvme-auth-wq] TS 1

0 255 19 0 [jbd2/nvme0n1p6-8] TS 1

0 256 39 -20 [ext4-rsv-conver] TS 1

0 295 20 -1 /lib/systemd/systemd-journa TS 4

0 366 19 0 /lib/systemd/systemd-udevd TS 4

0 414 39 -20 [cryptd] TS 1

0 423 39 -20 [led\_workqueue] TS 1

0 534 39 -20 [cfg80211] TS 1

0 543 39 -20 [kworker/u33:1-hci0] TS 1

0 567 39 -20 [amd\_iommu\_v2] TS 1

0 580 19 0 [kworker/10:2-cgroup\_destro TS 1

0 581 19 0 [napi/phy0-8193] TS 1

0 582 19 0 [napi/phy0-8194] TS 1

0 583 19 0 [napi/phy0-8195] TS 1

0 585 19 0 [jbd2/nvme0n1p7-8] TS 1

0 586 39 -20 [ext4-rsv-conver] TS 1

108 614 19 0 /lib/systemd/systemd-oomd TS 4

101 617 19 0 /lib/systemd/systemd-resolv TS 4

103 618 19 0 /lib/systemd/systemd-timesy TS 4

0 670 41 - [mt76-tx phy0] FF 1

0 672 39 -20 [amdgpu-reset-de] TS 1

0 674 19 0 /usr/libexec/accounts-daemo TS 4

0 675 19 0 /usr/sbin/acpid TS 4

114 677 19 0 avahi-daemon: running [katy TS 4

0 678 19 0 /usr/lib/bluetooth/bluetoot TS 4

0 679 19 0 /usr/sbin/cron -f -P TS 4

102 680 19 0 @dbus-daemon --system --add TS 4

0 681 19 0 /usr/sbin/NetworkManager -- TS 4

0 689 19 0 /usr/sbin/irqbalance --fore TS 4

0 691 19 0 /usr/bin/python3 /usr/bin/n TS 4

0 692 19 0 /usr/libexec/polkitd --no-d TS 4

0 693 19 0 /usr/libexec/power-profiles TS 4

104 694 19 0 /usr/sbin/rsyslogd -n -iNON TS 4

0 696 19 0 /usr/lib/snapd/snapd TS 4

0 697 19 0 /usr/libexec/switcheroo-con TS 4

0 698 19 0 /lib/systemd/systemd-logind TS 4

0 700 19 0 /usr/libexec/udisks2/udisks TS 4

0 701 19 0 /sbin/wpa\_supplicant -u -s TS 4

114 703 19 0 avahi-daemon: chroot helper TS 1

0 729 39 -20 [ttm] TS 1

0 770 39 -20 [amdgpu\_dm\_hpd\_r] TS 1

0 771 39 -20 [amdgpu\_dm\_hpd\_r] TS 1

0 772 39 -20 [dm\_vblank\_contr] TS 1

0 773 19 0 /usr/sbin/ModemManager TS 4

0 789 19 0 /usr/sbin/cupsd -l TS 4

0 792 19 0 /usr/bin/python3 /usr/share TS 4

0 799 90 - [card0-crtc0] FF 1

0 800 90 - [card0-crtc1] FF 1

0 801 90 - [card0-crtc2] FF 1

0 802 90 - [card0-crtc3] FF 1

0 803 41 - [gfx\_low] FF 1

0 804 41 - [gfx\_high] FF 1

0 805 41 - [comp\_1.0.0] FF 1

0 806 41 - [comp\_1.1.0] FF 1

0 807 41 - [comp\_1.2.0] FF 1

0 808 41 - [comp\_1.3.0] FF 1

0 809 41 - [comp\_1.0.1] FF 1

0 810 41 - [comp\_1.1.1] FF 1

0 811 41 - [comp\_1.2.1] FF 1

0 812 41 - [comp\_1.3.1] FF 1

0 813 41 - [sdma0] FF 1

0 814 41 - [vcn\_dec] FF 1

0 815 41 - [vcn\_enc0] FF 1

0 816 41 - [vcn\_enc1] FF 1

0 817 41 - [jpeg\_dec] FF 1

0 898 19 0 /usr/sbin/gdm3 TS 4

116 961 18 1 /usr/libexec/rtkit-daemon TS 4

0 1100 19 0 /usr/sbin/cups-browsed TS 4

113 1103 19 0 /usr/sbin/kerneloops --test TS 1

113 1105 19 0 /usr/sbin/kerneloops TS 1

0 1199 19 0 /usr/libexec/upowerd TS 4

0 1277 19 0 /usr/libexec/packagekitd TS 4

123 1409 19 0 /usr/libexec/colord TS 4

0 1493 19 0 gdm-session-worker [pam/gdm TS 4

1000 1509 19 0 /lib/systemd/systemd --user TS 4

1000 1510 19 0 (sd-pam) TS 5

1000 1516 30 -11 /usr/bin/pipewire TS 0

1000 1517 19 0 /usr/bin/pipewire-media-ses TS 0

1000 1518 30 -11 /usr/bin/pulseaudio --daemo TS 0

1000 1529 19 0 /usr/bin/dbus-daemon --sess TS 0

1000 1536 19 0 /usr/libexec/gvfsd TS 0

1000 1544 19 0 /usr/libexec/gvfsd-fuse /ru TS 0

1000 1559 19 0 /usr/libexec/xdg-document-p TS 0

1000 1564 19 0 /usr/libexec/xdg-permission TS 0

1000 1565 19 0 /usr/bin/gnome-keyring-daem TS 1

0 1573 19 0 fusermount3 -o rw,nosuid,no TS 4

1000 1584 0 - /usr/libexec/tracker-miner- IDL 0

0 1599 29 -10 [krfcommd] TS 5

1000 1600 19 0 /usr/libexec/gvfs-udisks2-v TS 0

1000 1605 19 0 /usr/libexec/gvfs-afc-volum TS 0

1000 1610 19 0 /usr/libexec/gvfs-goa-volum TS 0

1000 1614 19 0 /usr/libexec/goa-daemon TS 0

1000 1621 19 0 /usr/libexec/gdm-x-session TS 4

1000 1623 19 0 /usr/libexec/goa-identity-s TS 0

1000 1625 19 0 /usr/lib/xorg/Xorg vt2 -dis TS 4

1000 1627 19 0 /usr/libexec/gvfs-gphoto2-v TS 0

1000 1637 19 0 /usr/libexec/gvfs-mtp-volum TS 0

1000 1691 19 0 /usr/libexec/gnome-session- TS 0

1000 1770 19 0 /usr/libexec/at-spi-bus-lau TS 0

1000 1776 19 0 /usr/bin/dbus-daemon --conf TS 0

1000 1789 19 0 /usr/libexec/gnome-session- TS 0

1000 1801 19 0 /usr/libexec/gnome-session- TS 0

1000 1824 19 0 /usr/bin/gnome-shell TS 0

1000 1849 19 0 /usr/libexec/gnome-shell-ca TS 0

1000 1853 19 0 /usr/libexec/dconf-service TS 0

1000 1861 19 0 /usr/libexec/evolution-sour TS 0

1000 1862 19 0 /snap/snapd-desktop-integra TS 4

1000 1920 19 0 /snap/snapd-desktop-integra TS 1

1000 1924 19 0 /usr/libexec/gvfsd-trash -- TS 0

1000 1931 19 0 /usr/libexec/evolution-cale TS 0

1000 1941 19 0 /usr/bin/gjs /usr/share/gno TS 0

1000 1943 19 0 /usr/libexec/at-spi2-regist TS 0

1000 1967 19 0 /usr/libexec/evolution-addr TS 0

1000 1970 19 0 sh -c /usr/bin/ibus-daemon TS 0

1000 1972 19 0 /usr/libexec/gsd-a11y-setti TS 0

1000 1973 19 0 /usr/bin/ibus-daemon --pane TS 0

1000 1975 19 0 /usr/libexec/gsd-color TS 0

1000 1977 19 0 /usr/libexec/gsd-datetime TS 0

1000 1981 19 0 /usr/libexec/gsd-housekeepi TS 0

1000 1984 19 0 /usr/libexec/gsd-keyboard TS 0

1000 1986 19 0 /usr/libexec/gsd-media-keys TS 0

1000 1988 19 0 /usr/libexec/gsd-power TS 0

1000 1992 19 0 /usr/libexec/gsd-print-noti TS 0

1000 1995 19 0 /usr/libexec/gsd-rfkill TS 0

1000 1997 19 0 /usr/libexec/gsd-screensave TS 0

1000 2003 19 0 /usr/libexec/gsd-sharing TS 0

1000 2004 19 0 /usr/libexec/gsd-smartcard TS 0

1000 2007 19 0 /usr/libexec/gsd-sound TS 0

1000 2012 19 0 /usr/libexec/gsd-wacom TS 0

1000 2015 19 0 /usr/libexec/gsd-xsettings TS 0

1000 2031 19 0 /usr/libexec/evolution-data TS 0

1000 2034 19 0 /usr/libexec/gsd-disk-utili TS 0

1000 2038 19 0 /usr/libexec/ibus-dconf TS 0

1000 2047 19 0 /usr/libexec/ibus-extension TS 0

1000 2075 19 0 /usr/libexec/gsd-printer TS 0

1000 2108 19 0 /usr/libexec/ibus-x11 --kil TS 0

1000 2114 19 0 /usr/libexec/ibus-portal TS 0

1000 2121 19 0 /usr/bin/gjs /usr/share/gno TS 0

1000 2147 19 0 /usr/libexec/ibus-engine-si TS 0

1000 2156 19 0 /usr/libexec/xdg-desktop-po TS 0

1000 2173 19 0 /snap/snap-store/1113/usr/b TS 4

1000 2184 19 0 /usr/libexec/xdg-desktop-po TS 0

1000 2342 19 0 /usr/libexec/xdg-desktop-po TS 0

0 2368 39 -20 [kworker/11:3H-ttm] TS 1

0 2428 39 -20 [kworker/10:2H-ttm] TS 1

1000 2434 19 0 /usr/libexec/gvfsd-metadata TS 0

1000 2607 19 0 update-notifier TS 0

1000 2826 19 0 /opt/google/chrome/chrome TS 4

1000 2831 19 0 cat TS 0

1000 2832 19 0 cat TS 0

1000 2834 19 0 /opt/google/chrome/chrome\_c TS 0

1000 2836 19 0 /opt/google/chrome/chrome\_c TS 0

1000 2842 19 0 /opt/google/chrome/chrome - TS 0

1000 2843 19 0 /opt/google/chrome/chrome - TS 4

1000 2844 19 0 /snap/telegram-desktop/5820 TS 4

1000 2856 19 0 /opt/google/chrome/nacl\_hel TS 4

1000 2863 19 0 /opt/google/chrome/chrome - TS 5

1000 2965 19 0 /opt/google/chrome/chrome - TS 1

1000 2970 19 0 /opt/google/chrome/chrome - TS 0

1000 2980 19 0 /opt/google/chrome/chrome - TS 1

1000 3100 19 0 /opt/google/chrome/chrome - TS 1

1000 3111 19 0 /opt/google/chrome/chrome - TS 1

1000 3245 19 0 /opt/google/chrome/chrome - TS 1

1000 3966 19 0 /opt/google/chrome/chrome - TS 1

1000 4200 19 0 /opt/google/chrome/chrome - TS 0

1000 4216 19 0 /opt/google/chrome/chrome - TS 1

1000 4425 19 0 /usr/bin/nautilus --gapplic TS 0

1000 4573 19 0 /opt/google/chrome/chrome - TS 1

1000 4637 19 0 evince /home/katya/Загрузки TS 0

1000 4650 19 0 /usr/libexec/evinced TS 0

1000 4688 19 0 /usr/lib/libreoffice/progra TS 0

1000 4704 19 0 /usr/lib/libreoffice/progra TS 0

1000 5456 19 0 /snap/code/159/usr/share/co TS 0

1000 5458 19 0 /snap/code/159/usr/share/co TS 0

1000 5459 19 0 /snap/code/159/usr/share/co TS 0

1000 5472 19 0 /snap/code/159/usr/share/co TS 0

1000 5498 19 0 /snap/code/159/usr/share/co TS 0

1000 5520 19 0 /snap/code/159/usr/share/co TS 1

1000 5552 19 0 /snap/code/159/usr/share/co TS 0

1000 5593 19 0 /snap/code/159/usr/share/co TS 0

1000 5602 19 0 /snap/code/159/usr/share/co TS 0

1000 5603 19 0 /snap/code/159/usr/share/co TS 0

1000 5639 19 0 /snap/code/159/usr/share/co TS 0

1000 5680 19 0 /home/katya/.vscode/extensi TS 0

1000 5762 19 0 /usr/bin/bash --init-file / TS 0

0 6513 19 0 [kworker/7:2-events] TS 1

0 6901 19 0 [kworker/8:0-cgroup\_destroy TS 1

0 8719 39 -20 [kworker/5:2H-ttm] TS 1

0 8827 39 -20 [kworker/5:3H-kblockd] TS 1

1000 8915 19 0 /opt/google/chrome/chrome - TS 1

0 9151 19 0 [kworker/2:1-events] TS 1

0 9510 39 -20 [kworker/1:1H-ttm] TS 1

0 9512 39 -20 [kworker/1:4H-kblockd] TS 1

0 9553 39 -20 [kworker/0:2H-kblockd] TS 1

0 9555 39 -20 [kworker/4:2H-ttm] TS 1

1000 9612 19 0 /opt/google/chrome/chrome - TS 1

0 9700 19 0 [kworker/5:0-mm\_percpu\_wq] TS 1

1000 9720 19 0 /opt/google/chrome/chrome - TS 1

1000 9825 19 0 /home/katya/.vscode/extensi TS 0

0 10287 39 -20 [kworker/9:3H-kblockd] TS 1

0 10420 19 0 [kworker/u32:2-events\_power TS 1

0 10424 19 0 [kworker/1:0] TS 1

1000 10437 19 0 /home/katya/.vscode/extensi TS 0

0 11977 39 -20 [kworker/2:0H-kblockd] TS 1

0 11978 39 -20 [kworker/2:1H-ttm] TS 1

0 12014 39 -20 [kworker/8:2H-ttm] TS 1

0 12187 39 -20 [kworker/3:2H-ttm] TS 1

0 12248 19 0 [kworker/10:1-events\_freeza TS 1

1000 12298 19 0 gjs /usr/share/gnome-shell/ TS 0

0 12372 19 0 [kworker/3:0-events] TS 1

0 12373 19 0 [kworker/4:0-events] TS 1

0 12376 19 0 [kworker/2:2] TS 1

0 12377 19 0 [kworker/8:2] TS 1

0 12379 19 0 [kworker/6:1-events] TS 1

0 12418 19 0 [kworker/7:0] TS 1

0 12419 39 -20 [kworker/6:1H-kblockd] TS 1

0 12429 39 -20 [kworker/9:1H-ttm] TS 1

1000 12511 19 0 /home/katya/.vscode/extensi TS 0

0 12590 19 0 [kworker/11:1-events] TS 1

0 12781 39 -20 [kworker/0:0H] TS 1

0 12849 19 0 [kworker/3:1-events] TS 1

0 12880 39 -20 [kworker/3:0H-kblockd] TS 1

0 12986 19 0 [kworker/u32:4+events\_unbou TS 1

0 13099 19 0 [kworker/0:2-events] TS 1

1000 13388 19 0 /usr/bin/gedit --gapplicati TS 0

0 13632 19 0 [kworker/u32:5-events\_power TS 1

0 13663 19 0 [kworker/0:0] TS 1

0 13664 19 0 [kworker/11:0] TS 1

1000 13846 19 0 /usr/libexec/gnome-terminal TS 0

1000 13877 19 0 bash TS 0

0 13900 19 0 [kworker/4:2-events] TS 1

0 13986 19 0 [kworker/9:2-events] TS 1

0 13991 39 -20 [kworker/8:1H-kblockd] TS 1

0 13992 39 -20 [kworker/4:0H-ttm] TS 1

0 14054 39 -20 [kworker/11:0H-kblockd] TS 1

0 14055 19 0 [kworker/u32:0-events\_unbou TS 1

0 14056 39 -20 [kworker/10:0H] TS 1

0 14064 19 0 [kworker/8:1-events] TS 1

0 14066 19 0 [kworker/9:0] TS 1

0 14067 19 0 [kworker/u32:1] TS 1

0 14068 19 0 [kworker/5:2] TS 1

0 14078 39 -20 [kworker/4:1H-kblockd] TS 1

1000 14088 19 0 /opt/google/chrome/chrome - TS 1

1000 14134 19 0 ps -o uid,pid,pri,ni,cmd,cl TS 4

**Задание 18.** *Многонитевое функционирование*

Подготовьте программу, формирующую несколько нитей. Нити для эксперимента могут быть практически идентичны.

Например, каждая нить в цикле: выводит на печать собственное имя и инкрементирует переменную времени, после чего "засыпает" (sleep(5); sleep(1); -для первой и второй нитей соответственно), на экран (в файл) должно выводиться имя нити и количество пятисекундных (для первой) и секундных (для второй) интервалов функционирования каждой нити.

(base) katya@katya:~/os/lb4$ cat **task18.c**

#include <signal.h>

#include <pthread.h>

#include <stdio.h>

#include <time.h>

#include <unistd.h>

#include <stdlib.h>

//тип pthread\_t используется для хранения идентификаторов нитей выполнения

pthread\_t t1, t2;

void\* thread1(void\*){

printf("Thread\_1 start\n");

system("ps -ft >> th1.txt");

//фиксируем время начала

time\_t start = time(NULL);

int n = 0;

for(int i = 0; i<5; i++){

//увеличиваем счетчик и фиксируем время работы

time\_t end = time(NULL);

printf("th1 num %d, time=%ld\n", ++n, end-start);

sleep(5);

}

printf("Thread\_1 finish\n");

}

void\* thread2(void\*){

printf("Thread\_2 start\n");

system("ps -ft >> th2.txt");

//фиксируем время начала

time\_t start = time(NULL);

int n = 0;

for(int i = 0; i<10; i++){

//увеличиваем счетчик и фиксируем время работы

time\_t end = time(NULL);

printf("th2 num %d, time=%ld\n", ++n, end-start);

sleep(1);

}

printf("Thread\_2 finish\n");

}

int main(){

system(">th1.txt");

system(">th2.txt");

system("ps -ft > 18res.txt");

//запускаем две нити, их tid записываем в t1 b t2

pthread\_create(&t1, NULL, thread1, NULL);

pthread\_create(&t2, NULL, thread2, NULL);

//ждем выполнения нитей

pthread\_join(t1, NULL);

pthread\_join(t2, NULL);

system("ps -ft >> 18res.txt");

return 0;

}

(base) katya@katya:~/os/lb4$ gcc task18.c -o task18

(base) katya@katya:~/os/lb4$ ./task18

Thread\_1 start

Thread\_2 start

th1 num 1, time=0

th2 num 1, time=0

th2 num 2, time=1

th2 num 3, time=2

th2 num 4, time=3

th2 num 5, time=4

th1 num 2, time=5

th2 num 6, time=5

th2 num 7, time=6

th2 num 8, time=7

th2 num 9, time=8

th2 num 10, time=9

th1 num 3, time=10

Thread\_2 finish

th1 num 4, time=15

th1 num 5, time=20

Thread\_1 finish

(base) katya@katya:~/os/lb4$ cat 18res.txt

PID TTY STAT TIME COMMAND

6329 pts/2 Ss 0:00 bash

6353 pts/2 S+ 0:00 \\_ ./task18

6357 pts/2 S+ 0:00 \\_ sh -c ps -ft > 18res.txt

6358 pts/2 R+ 0:00 \\_ ps -ft

PID TTY STAT TIME COMMAND

6329 pts/2 Ss 0:00 bash

6353 pts/2 S+ 0:00 \\_ ./task18

6366 pts/2 S+ 0:00 \\_ sh -c ps -ft >> 18res.txt

6367 pts/2 R+ 0:00 \\_ ps -ft

(base) katya@katya:~/os/lb4$ cat th1.txt

PID TTY STAT TIME COMMAND

6329 pts/2 Ss 0:00 bash

6353 pts/2 Sl+ 0:00 \\_ ./task18

6361 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th1.txt

6364 pts/2 R+ 0:00 | \\_ ps -ft

6362 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th2.txt

6363 pts/2 R+ 0:00 \\_ ps -ft

(base) katya@katya:~/os/lb4$ cat th2.txt

PID TTY STAT TIME COMMAND

6329 pts/2 Ss 0:00 bash

6353 pts/2 Sl+ 0:00 \\_ ./task18

6361 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th1.txt

6364 pts/2 R+ 0:00 | \\_ ps -ft

6362 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th2.txt

6363 pts/2 R+ 0:00 \\_ ps -ft

В ходе данной работы первая нить спала 5 раз по 10 сек, а вторая - 10 раз по 1 сек. На экран происходил вывод текущего потока, номера итерации и время работы.

*pthread\_t* — это тип данных, используемый для уникального идентификации нити.

Функция *pthread\_create* используется для создания новой нити в программе. Она принимает следующие параметры:

* pthread\_t \*thread\_id: Указатель на переменную, в которой будет храниться идентификатор созданной нити.
* const pthread\_attr\_t \*attr: Атрибуты нити, которые могут быть установлены для изменения поведения нити.
* void \*(\*start\_routine)(void \*): Функция, которая будет выполняться в новой нити.
* void \*arg: Аргумент, который будет передан в функцию start\_routine.

Функция *pthread\_join* позволяет главной нити ожидать завершения целевой нити. Она блокирует выполнение главной нити до тех пор, пока целевая нить не завершится.

Взаимодействие нитей: Нити работали независимо друг от друга, каждый выполняя свой собственный цикл с заданными интервалами сна.

Завершение нитей: Thread\_2 завершилась раньше, чем Thread\_1, что связано с различием в интервалах сна.

Производительность: Время выполнения каждого шага зависит от заданного интервала сна, что позволяет анализировать, как различные задержки влияют на общую производительность и распределение времени между нитями.

**Задание 19.** После запуска программы проанализируйте выполнение нитей, распределение во времени. Используйте для этого вывод таблицы процессов командой ps -axhf. Попробуйте удалить нить, зная ее идентификатор, командой kill. Приведите и объясните результат.

Программа из 18 задания была модифицирована:

(base) katya@katya:~/os/lb4$ cat **task19.c**

#include <signal.h>

#include <pthread.h>

#include <stdio.h>

#include <time.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/syscall.h>

//тип pthread\_t используется для хранения идентификаторов нитей выполнения

pthread\_t t1, t2;

void\* thread1(void\*){

printf("Thread\_1 start\n");

system("ps -ft >> th1\_19.txt");

//фиксируем время начала

time\_t start = time(NULL);

int n = 0;

for(int i = 0; i<5; i++){

//увеличиваем счетчик и фиксируем время работы

time\_t end = time(NULL);

printf("th1 num %d, time=%ld\n", ++n, end-start);

sleep(5);

//убийство нити по ее tid

kill(syscall(SYS\_gettid), SIGINT);

}

printf("Thread\_1 finish\n");

}

void\* thread2(void\*){

printf("Thread\_2 start\n");

system("ps -ft >> th2\_19.txt");

//фиксируем время начала

time\_t start = time(NULL);

int n = 0;

for(int i = 0; i<10; i++){

//увеличиваем счетчик и фиксируем время работы

time\_t end = time(NULL);

printf("th2 num %d, time=%ld\n", ++n, end-start);

sleep(1);

}

printf("Thread\_2 finish\n");

}

int main(){

system(">th1\_19.txt");

system(">th2\_19.txt");

system("ps -ft > 19res.txt");

//запускаем две нити, их tid записываем в t1 b t2

pthread\_create(&t1, NULL, thread1, NULL);

pthread\_create(&t2, NULL, thread2, NULL);

//ждем выполнения нитей

pthread\_join(t1, NULL);

pthread\_join(t2, NULL);

system("ps -ft >> 19res.txt");

return 0;

}

(base) katya@katya:~/os/lb4$ gcc task19.c -o task19

(base) katya@katya:~/os/lb4$ ./task19

Thread\_1 start

Thread\_2 start

th2 num 1, time=0

th1 num 1, time=0

th2 num 2, time=1

th2 num 3, time=2

th2 num 4, time=3

th2 num 5, time=4

(base) katya@katya:~/os/lb4$ cat 19res.txt

PID TTY STAT TIME COMMAND

6329 pts/2 Ss 0:00 bash

6804 pts/2 S+ 0:00 \\_ ./task19

6810 pts/2 S+ 0:00 \\_ sh -c ps -ft > 19res.txt

6811 pts/2 R+ 0:00 \\_ ps -ft

(base) katya@katya:~/os/lb4$ cat th1\_19.txt

PID TTY STAT TIME COMMAND

6329 pts/2 Ss 0:00 bash

6804 pts/2 Sl+ 0:00 \\_ ./task19

6814 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th1\_19.txt

6817 pts/2 R+ 0:00 | \\_ ps -ft

6815 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th2\_19.txt

6816 pts/2 R+ 0:00 \\_ ps -ft

(base) katya@katya:~/os/lb4$ cat th2\_19.txt

PID TTY STAT TIME COMMAND

6329 pts/2 Ss 0:00 bash

6804 pts/2 Sl+ 0:00 \\_ ./task19

6814 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th1\_19.txt

6817 pts/2 R+ 0:00 | \\_ ps -ft

6815 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th2\_19.txt

6816 pts/2 R+ 0:00 \\_ ps -ft

Видно, что отправив kill по tid нити, завершился процесс в целом, чего и следовало ожидать, так как нужно использовать другие функции и устанавливать обработчик сигнала: по умолчанию многие сигналы приводят к завершению программы.

**Задание 20.** Модифицируйте программу так, чтобы управление второй нитью осуществлялось посредством сигнала SIGUSR1 из первой нити. На пятой секунде работы приложения удалите вторую нить. Для этого воспользуйтесь функцией pthread\_kill(t2, SIGUSR); (t2 - дескриптор второй нити). В остальном программу можно не изменять. Проанализируйте полученные результаты.

Функция pthread\_kill используется для асинхронной доставки сигнала конкретной нити в процессе, который вызывает эту функцию. Это позволяет, например, одному потоку воздействовать на распространение сигнала среди набора потоков. Однако важно отметить, что действие сигнала (терминирование или остановка) влияет на весь процесс, а не только на конкретную нить, к которой был отправлен сигнал.

Основные моменты использования pthread\_kill:

1) Отправка сигнала: Функция pthread\_kill принимает два параметра: идентификатор нити (pthread\_t thread) и сигнал (int sig). Сигнал может быть любым из стандартных сигналов, таких как SIGTERM или SIGKILL, или пользовательским сигналом, таким как SIGUSR1 или SIGUSR2.

2) Обработка сигнала: Если сигнал, переданный через pthread\_kill, не обрабатывается в нити, которая его получила, то поведение зависит от типа сигнала.

3) Ошибка: Если сигнал равен нулю (sig == 0), функция pthread\_kill выполняет проверку ошибок, но не отправляет никакого сигнала. Это полезно для проверки существования нити без отправки сигнала.

(base) katya@katya:~/os/lb4$ cat **task20.c**

#include <signal.h>

#include <pthread.h>

#include <stdio.h>

#include <time.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/syscall.h>

//тип pthread\_t используется для хранения идентификаторов нитей выполнения

pthread\_t t1, t2;

void\* thread1(void\*){

printf("Thread\_1 start\n");

system("ps -ft >> th1\_20.txt");

//фиксируем время начала

time\_t start = time(NULL);

int n = 0;

for(int i = 0; i<5; i++){

//увеличиваем счетчик и фиксируем время работы

time\_t end = time(NULL);

printf("th1 num %d, time=%ld\n", ++n, end-start);

sleep(5);

//убийство второй нити по ее дескриптору

pthread\_kill(t2, SIGUSR1);

}

printf("Thread\_1 finish\n");

}

void\* thread2(void\*){

printf("Thread\_2 start\n");

system("ps -ft >> th2\_20.txt");

//фиксируем время начала

time\_t start = time(NULL);

int n = 0;

for(int i = 0; i<10; i++){

//увеличиваем счетчик и фиксируем время работы

time\_t end = time(NULL);

printf("th2 num %d, time=%ld\n", ++n, end-start);

sleep(1);

}

printf("Thread\_2 finish\n");

}

int main(){

system(">th1\_20.txt");

system(">th2\_20.txt");

system("ps -ft > 20res.txt");

//запускаем две нити, их tid записываем в t1 b t2

pthread\_create(&t1, NULL, thread1, NULL);

pthread\_create(&t2, NULL, thread2, NULL);

sleep(2);

//ждем выполнения нитей

pthread\_join(t1, NULL);

pthread\_join(t2, NULL);

system("ps -ft >> 20res.txt");

return 0;

}

(base) katya@katya:~/os/lb4$ gcc task20.c -o task20

(base) katya@katya:~/os/lb4$ ./task20

Thread\_1 start

Thread\_2 start

th2 num 1, time=0

th1 num 1, time=0

th2 num 2, time=1

th2 num 3, time=2

th2 num 4, time=3

th2 num 5, time=4

Определяемый пользователем сигнал 1

(base) katya@katya:~/os/lb4$ cat 20res.txt

PID TTY STAT TIME COMMAND

7741 pts/2 Ss 0:00 bash

8387 pts/2 S+ 0:00 \\_ ./task20

8392 pts/2 S+ 0:00 \\_ sh -c ps -ft > 20res.txt

8393 pts/2 R+ 0:00 \\_ ps -ft

(base) katya@katya:~/os/lb4$ cat th1\_20.txt

PID TTY STAT TIME COMMAND

7741 pts/2 Ss 0:00 bash

8387 pts/2 Sl+ 0:00 \\_ ./task20

8396 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th1\_20.txt

8399 pts/2 R+ 0:00 | \\_ ps -ft

8397 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th2\_20.txt

8398 pts/2 R+ 0:00 \\_ ps -ft

(base) katya@katya:~/os/lb4$ cat th2\_20.txt

PID TTY STAT TIME COMMAND

7741 pts/2 Ss 0:00 bash

8387 pts/2 Sl+ 0:00 \\_ ./task20

8396 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th1\_20.txt

8399 pts/2 R+ 0:00 | \\_ ps -ft

8397 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th2\_20.txt

8398 pts/2 R+ 0:00 \\_ ps -ft

Видно, что, как и в прошлый раз, процесс завершился целиком, так как не было установлено обработчика.

**Задание 21.** Последняя модификация предполагает создание собственного обработчика сигнала, содержащего уведомление о начале его работы и возврат посредством функции pthread\_exit(NULL); Сравните результаты, полученные после запуска этой модификации программы с результатами предыдущей

(base) katya@katya:~/os/lb4$ cat task21.c

#include <signal.h>

#include <pthread.h>

#include <stdio.h>

#include <time.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/syscall.h>

//тип pthread\_t используется для хранения идентификаторов нитей выполнения

pthread\_t t1, t2;

//завершаем поток при помощи pthread\_exit и выводим инфу о потоке

void handler(){

printf("сигнал принялся tid=%ld\n", syscall(SYS\_gettid));

pthread\_exit(NULL);

}

void\* thread1(void\*){

printf("Thread\_1 start\n");

system("ps -ft >> th1\_21.txt");

//фиксируем время начала

time\_t start = time(NULL);

int n = 0;

for(int i = 0; i<5; i++){

//увеличиваем счетчик и фиксируем время работы

time\_t end = time(NULL);

printf("th1 num %d, time=%ld\n", ++n, end-start);

sleep(5);

//убийство второй нити по ее дескриптору на первой итерации

if(i==0){

printf("сигнал отправился из tid=%ld\n", syscall(SYS\_gettid));

pthread\_kill(t2, SIGUSR1);

}

}

printf("Thread\_1 finish\n");

}

void\* thread2(void\*){

printf("Thread\_2 start\n");

system("ps -ft >> th2\_21.txt");

//фиксируем время начала

time\_t start = time(NULL);

int n = 0;

for(int i = 0; i<10; i++){

//увеличиваем счетчик и фиксируем время работы

time\_t end = time(NULL);

printf("th2 num %d, time=%ld\n", ++n, end-start);

sleep(1);

}

printf("Thread\_2 finish\n");

}

int main(){

system(">th1\_21.txt");

system(">th2\_21.txt");

system("ps -ft > 21res.txt");

//установка обработчика для сигнала

signal(SIGUSR1, handler);

//запускаем две нити, их tid записываем в t1 & t2

pthread\_create(&t1, NULL, thread1, NULL);

pthread\_create(&t2, NULL, thread2, NULL);

sleep(2);

//ждем выполнения нитей

pthread\_join(t1, NULL);

pthread\_join(t2, NULL);

system("ps -ft >> 21res.txt");

return 0;

}

(base) katya@katya:~/os/lb4$ gcc task21.c -o task21

(base) katya@katya:~/os/lb4$ ./task21

Thread\_1 start

Thread\_2 start

th2 num 1, time=0

th1 num 1, time=0

th2 num 2, time=1

th2 num 3, time=2

th2 num 4, time=3

th2 num 5, time=4

сигнал отправился из tid=9622

th1 num 2, time=5

сигнал принялся tid=9623

th1 num 3, time=10

th1 num 4, time=15

th1 num 5, time=20

Thread\_1 finish

(base) katya@katya:~/os/lb4$ cat 21res.txt

PID TTY STAT TIME COMMAND

7741 pts/2 Ss 0:00 bash

9613 pts/2 S+ 0:00 \\_ ./task21

9619 pts/2 S+ 0:00 \\_ sh -c ps -ft > 21res.txt

9621 pts/2 R+ 0:00 \\_ ps -ft

PID TTY STAT TIME COMMAND

7741 pts/2 Ss 0:00 bash

9613 pts/2 S+ 0:00 \\_ ./task21

9634 pts/2 S+ 0:00 \\_ sh -c ps -ft >> 21res.txt

9635 pts/2 R+ 0:00 \\_ ps -ft

(base) katya@katya:~/os/lb4$ cat th1\_21.txt

PID TTY STAT TIME COMMAND

7741 pts/2 Ss 0:00 bash

9613 pts/2 Sl+ 0:00 \\_ ./task21

9624 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th1\_21.txt

9626 pts/2 R+ 0:00 | \\_ ps -ft

9625 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th2\_21.txt

9627 pts/2 R+ 0:00 \\_ ps -ft

(base) katya@katya:~/os/lb4$ cat th2\_21.txt

PID TTY STAT TIME COMMAND

7741 pts/2 Ss 0:00 bash

9613 pts/2 Sl+ 0:00 \\_ ./task21

9624 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th1\_21.txt

9626 pts/2 R+ 0:00 | \\_ ps -ft

9625 pts/2 S+ 0:00 \\_ sh -c ps -ft >> th2\_21.txt

9627 pts/2 R+ 0:00 \\_ ps -ft

Таким образом, при установке обработчика с использованием функции pthread\_exit завершился только первый поток, при этом процесс не завершился в целом и не создалось никаких “помех” для второго потока.

**Задание 22.** Перехватите сигнал «CTRL C» для процесса и потока однократно, а также многократно с восстановлением исходного обработчика после нескольких раз срабатывания. Проделайте аналогичную работу для переназначения другой комбинации клавиш.

Установим обработчик для сигнала SIGINT (CTRL C), видно, что сигнал корректно обрабатывается:

(base) katya@katya:~/os/lb4$ cat **task22.c**

#include <signal.h>

#include <pthread.h>

#include <stdio.h>

#include <time.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/syscall.h>

//кол-во вызовов обработчика

int count = 0;

//идентификатор нити

pthread\_t t1;

void handler(){

count += 1;

//вывод потока, обрабатывающего сигнал

printf("сигнал принимается tid=%ld\n", syscall(SYS\_gettid));

//восстановление обработчика

if(count == 1)

// 1 - максимальное кол-во вызовов до восстановления обработчика по умолчанию

signal(SIGINT, SIG\_DFL);

}

void\* thread1(void\*){

printf("Thread\_1 start tid=%ld\n", syscall(SYS\_gettid));

system("ps -ft >> th1\_22.txt");

//фиксируем время начала

time\_t start = time(NULL);

int n = 0;

for(int i = 0; i<4; i++){

//увеличиваем счетчик и фиксируем время работы

time\_t end = time(NULL);

printf("th1 num %d, time=%ld\n", ++n, end-start);

sleep(5);

}

printf("Thread\_1 finish\n");

}

int main(){

system(">th1\_22.txt");

system("ps -ft > 22res.txt");

//установка обработчика для сигнала

signal(SIGINT, handler);

printf("главная нить tid=%ld\n", syscall(SYS\_gettid));

//запускаем нить

pthread\_create(&t1, NULL, thread1, NULL);

sleep(2);

system("ps -ft >> 22res.txt");

//ждем выполнения нити

pthread\_join(t1, NULL);

system("ps -ft >> 22res.txt");

return 0;

}

Изменим сигнал на SIGTSTP (CTRL Z), тоже все обрабатывается корректно:

(base) katya@katya:~/os/lb4$ gcc task22.c -o task22

(base) katya@katya:~/os/lb4$ ./task22

главная нить tid=12180

Thread\_1 start tid=12191

th1 num 1, time=0

^Cсигнал принимается tid=12180

th1 num 2, time=5

^C

(base) katya@katya:~/os/lb4$ ./task22

главная нить tid=12396

Thread\_1 start tid=12407

th1 num 1, time=0

^Zсигнал принимается tid=12396

th1 num 2, time=5

th1 num 3, time=10

^Z

[1]+ Остановлен ./task22

(base) katya@katya:~/os/lb4$ fg %1

./task22

th1 num 4, time=72

Thread\_1 finish

В случае перехвата для созданной нити потребовалось заблокировать основному потоку через изменение маски обработку нужного сигнала, а внутри созданного потока разблокировать ее. Это достигалось использованием набора функций, основной из них является sigprocmask. По итогу нужный результат был достигнут и при посылке сигнала процессу вызывался обработчик не основного потока, а созданной нити. Всё сработало корректно.

(base) katya@katya:~/os/lb4$ cat **task22\_1.c**

#include <signal.h>

#include <pthread.h>

#include <stdio.h>

#include <time.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/syscall.h>

//кол-во вызовов обработчика

int count = 0;

//идентификатор нити

pthread\_t t1;

void handler(){

count += 1;

//вывод потока, обрабатывающего сигнал

printf("сигнал принимается tid=%ld\n", syscall(SYS\_gettid));

//восстанавление обработчика

if(count == 3)

// 3 - максимальное кол-во вызовов до восстановления обработчика по умолчанию

signal(SIGINT, SIG\_DFL);

}

void\* thread1(void\*){

sigset\_t set;

sigemptyset(&set);

sigaddset(&set, SIGINT);

sigprocmask(SIG\_UNBLOCK, &set, NULL);

signal(SIGINT, handler);

printf("Thread\_1 start tid=%ld\n", syscall(SYS\_gettid));

system("ps -ft >> th1\_22.txt");

//фиксируем время начала

time\_t start = time(NULL);

int n = 0;

for(int i = 0; i<4; i++){

//увеличиваем счетчик и фиксируем время работы

time\_t end = time(NULL);

printf("th1 num %d, time=%ld\n", ++n, end-start);

sleep(5);

}

printf("Thread\_1 finish\n");

}

int main(){

system(">th1\_22.txt");

system("ps -ft > 22res.txt");

//установка обработчика для сигнала

sigset\_t set;

sigemptyset(&set);

sigaddset(&set, SIGINT);

sigprocmask(SIG\_BLOCK, &set, NULL);

printf("главная нить tid=%ld\n", syscall(SYS\_gettid));

//запускаем нить

pthread\_create(&t1, NULL, thread1, NULL);

sleep(2);

system("ps -ft >> 22res.txt");

//ждем выполнения нити

pthread\_join(t1, NULL);

system("ps -ft >> 22res.txt");

return 0;

}

(base) katya@katya:~/os/lb4$ gcc task22\_1.c -o task22\_1

(base) katya@katya:~/os/lb4$ ./task22\_1

главная нить tid=13081

Thread\_1 start tid=13093

th1 num 1, time=0

^Cсигнал принимается tid=13093

th1 num 2, time=1

^Cсигнал принимается tid=13093

th1 num 3, time=1

^Cсигнал принимается tid=13093

th1 num 4, time=2

^C

**Задание 23.** С помощью утилиты kill выведите список всех сигналов и дайте их краткую характеристику на основе документации ОС. Для чего предназначены сигналы с 32 по 64-й. Приведите пример их применения.

(base) katya@katya:~/os/lb4$ kill -l

1) SIGHUP 2) SIGINT 3) SIGQUIT 4) SIGILL 5) SIGTRAP

6) SIGABRT 7) SIGBUS 8) SIGFPE 9) SIGKILL 10) SIGUSR1

11) SIGSEGV 12) SIGUSR2 13) SIGPIPE 14) SIGALRM 15) SIGTERM

16) SIGSTKFLT 17) SIGCHLD 18) SIGCONT 19) SIGSTOP 20) SIGTSTP

21) SIGTTIN 22) SIGTTOU 23) SIGURG 24) SIGXCPU 25) SIGXFSZ

26) SIGVTALRM 27) SIGPROF 28) SIGWINCH 29) SIGIO 30) SIGPWR

31) SIGSYS 34) SIGRTMIN 35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3

38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8

43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13

48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12

53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9 56) SIGRTMAX-8 57) SIGRTMAX-7

58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2

63) SIGRTMAX-1 64) SIGRTMAX

Можно использовать команду man 7 signal для просмотра страницы руководства по сигналам, которая предоставляет детальную информацию о каждом сигнале, включая их назначение и поведение.

SIGABRT P1990 Core Abort signal from abort(3)

SIGALRM P1990 Term Timer signal from alarm(2)

SIGBUS P2001 Core Bus error (bad memory access)

SIGCHLD P1990 Ign Child stopped or terminated

SIGCLD - Ign A synonym for SIGCHLD

SIGCONT P1990 Cont Continue if stopped

SIGEMT - Term Emulator trap

SIGFPE P1990 Core Floating-point exception

SIGHUP P1990 Term Hangup detected on controlling terminal

or death of controlling process

SIGILL P1990 Core Illegal Instruction

SIGINFO - A synonym for SIGPWR

SIGINT P1990 Term Interrupt from keyboard

SIGIO - Term I/O now possible (4.2BSD)

SIGIOT - Core IOT trap. A synonym for SIGABRT

SIGKILL P1990 Term Kill signal

SIGLOST - Term File lock lost (unused)

SIGPIPE P1990 Term Broken pipe: write to pipe with no

readers; see pipe(7)

SIGPOLL P2001 Term Pollable event (Sys V);

synonym for SIGIO

SIGPROF P2001 Term Profiling timer expired

SIGPWR - Term Power failure (System V)

SIGQUIT P1990 Core Quit from keyboard

SIGSEGV P1990 Core Invalid memory reference

SIGSTKFLT - Term Stack fault on coprocessor (unused)

SIGSTOP P1990 Stop Stop process

SIGTSTP P1990 Stop Stop typed at terminal

SIGSYS P2001 Core Bad system call (SVr4);

see also seccomp(2)

SIGTERM P1990 Term Termination signal

SIGTRAP P2001 Core Trace/breakpoint trap

SIGTTIN P1990 Stop Terminal input for background process

SIGTTOU P1990 Stop Terminal output for background process

SIGUNUSED - Core Synonymous with SIGSYS

SIGURG P2001 Ign Urgent condition on socket (4.2BSD)

SIGUSR1 P1990 Term User-defined signal 1

SIGUSR2 P1990 Term User-defined signal 2

SIGVTALRM P2001 Term Virtual alarm clock (4.2BSD)

SIGXCPU P2001 Core CPU time limit exceeded (4.2BSD);

see setrlimit(2)

SIGXFSZ P2001 Core File size limit exceeded (4.2BSD);

see setrlimit(2)

SIGWINCH - Ign Window resize signal (4.3BSD, Sun)

Сигналы реального времени могут принимать значения между SIGRTMIN и SIGRTMAX включительно. POSIX требует, чтобы предоставлялось по крайней мере RTSIG\_MAX сигналов, и минимальное значение этой константы равно 8. Сигналы реального времени, в отличие от обычных, выстраиваются в очередь и обрабатываются, не сливаясь.

(base) katya@katya:~/os/lb4$ cat task23.c

#include <signal.h>

#include <pthread.h>

#include <stdio.h>

#include <time.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/syscall.h>

//идентификатор нити

pthread\_t t1, t2;

int pid;

void handler(int signum, siginfo\_t \*info, void \*context){

char tmp[100];

sprintf(tmp, "echo \"Перехвачен сигнал SIGRTMIN+%d\" >> 23res.txt", info->si\_value.sival\_int);

system(tmp);

}

// ОТПРАВКА СИГНАЛОВ РЕАЛ ВРЕМЕНИ С ДОП ДАННЫМИ

void\* thread1(void\*){

union sigval value;

value.sival\_int = 0;

for(int i=0; i<20; i++)

sigqueue(pid, SIGRTMIN, value);

while (1){}

}

void\* thread2(void\*){

union sigval value;

value.sival\_int = 1;

for(int i=0; i<20; i++)

sigqueue(pid, SIGRTMIN+1, value);

while (1){}

}

int main(){

system(">res23.txt");

//фиксируем pid, чтобы из нити слать сигналы

pid = getpid();

// установка обработчиков для сигналов реального времени

struct sigaction sa;

sa.sa\_flags = SA\_SIGINFO;

sa.sa\_sigaction = handler;

sigemptyset(&sa.sa\_mask);

sigaction(SIGRTMIN, &sa, NULL);

sigaction(SIGRTMIN+1, &sa, NULL);

//запускаем нити

pthread\_create(&t1, NULL, thread1, NULL);

pthread\_create(&t2, NULL, thread2, NULL);

while(1){}

return 0;

}

(base) katya@katya:~/os/lb4$ gcc task23.c -o task23

(base) katya@katya:~/os/lb4$ ./task23

^C

(base) katya@katya:~/os/lb4$ cat 23res.txt

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+1

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Перехвачен сигнал SIGRTMIN+0

Вывод программы указывает на то, что она успешно перехватила и обработала сигналы реального времени (SIGRTMIN и SIGRTMIN+1). Сообщения в файле подтверждают, что сигналы были отправлены и получены в соответствии с ожиданиями.

Количество записей в файле соответствует количеству отправленных сигналов, то есть 20 записей для каждого типа сигнала (SIGRTMIN и SIGRTMIN+1), что подтверждает, что сигналы были отправлены и получены в цикле.

Таким образом, вывод программы подтверждает, что она корректно отправляла сигналы реального времени из двух нитей и успешно перехватывала их, записывая информацию о перехваченных сигналах в файл.

**Задание 24.** Проанализируйте процедуру планирования для процессов и потоков одного процесса.

24.1. Обоснуйте результат экспериментально.

24.2. Попробуйте процедуру планирования изменить. Подтвердите экспериментально, если изменение возможно.

24.3. Задайте нитям разные приоритеты программно и извне (объясните результат).

(base) katya@katya:~/os/lb4$ cat **task24\_1.c**

#include <signal.h>

#include <pthread.h>

#include <stdio.h>

#include <time.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/syscall.h>

pthread\_t t1, t2, t3, t4;

void work(){

int n = 0;

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 1000000000; j++)

n += 1;

printf("tid=%ld\n", syscall(SYS\_gettid));

}

}

void\* thread1(void\*) {

printf("START: tid=%ld\n", syscall(SYS\_gettid));

work();

printf("END: tid=%ld\n", syscall(SYS\_gettid));

}

void\* thread2(void\*) {

printf("START: tid=%ld\n", syscall(SYS\_gettid));

work();

printf("END: tid=%ld\n", syscall(SYS\_gettid));

}

void\* thread3(void\*) {

printf("START: tid=%ld\n", syscall(SYS\_gettid));

work();

printf("END: tid=%ld\n", syscall(SYS\_gettid));

}

void\* thread4(void\*) {

printf("START: tid=%ld\n", syscall(SYS\_gettid));

work();

printf("END: tid=%ld\n", syscall(SYS\_gettid));

}

int main()

{

// меняем политику планирования родителя

// struct sched\_param param;

// param.sched\_priority = 1;

// sched\_setscheduler(0, SCHED\_FIFO, &param);

switch (sched\_getscheduler(0))

{

case SCHED\_FIFO:

printf("SCHED\_FIFO\n");

break;

case SCHED\_RR:

printf("SCHED\_RR\n");

break;

case SCHED\_OTHER:

printf("SCHED\_OTHER\n");

break;

}

// запускаем потоки

pthread\_create(&t1, NULL, thread1, NULL);

pthread\_create(&t2, NULL, thread2, NULL);

pthread\_create(&t3, NULL, thread3, NULL);

pthread\_create(&t4, NULL, thread4, NULL);

system("ps -eLo pid,tid,cls,pri,rtprio,ni,cmd | grep ./task24\_1 > 24\_1res.txt");

pthread\_join(t1, NULL);

pthread\_join(t2, NULL);

pthread\_join(t3, NULL);

pthread\_join(t4, NULL);

return 0;

}(base) katya@katya:~/os/lb4$ gcc task24\_1.c -o task24\_1

(base) katya@katya:~/os/lb4$ taskset 1 ./task24\_1

SCHED\_OTHER

START: tid=17583

START: tid=17582

START: tid=17581

START: tid=17580

tid=17581

tid=17582

tid=17583

tid=17580

tid=17583

tid=17581

tid=17582

tid=17580

tid=17581

tid=17582

tid=17583

tid=17580

tid=17583

tid=17581

tid=17582

tid=17580

tid=17581

tid=17582

tid=17583

tid=17580

tid=17583

tid=17581

tid=17582

tid=17580

tid=17581

tid=17582

tid=17583

tid=17580

tid=17582

tid=17583

tid=17581

tid=17580

tid=17582

tid=17581

tid=17583

tid=17580

tid=17581

END: tid=17581

tid=17583

END: tid=17583

tid=17580

END: tid=17580

tid=17582

END: tid=17582

(base) katya@katya:~/os/lb4$ cat 24\_1res.txt

17579 17579 TS 19 - 0 ./task24\_1

17579 17580 TS 19 - 0 ./task24\_1

17579 17581 TS 19 - 0 ./task24\_1

17579 17582 TS 19 - 0 ./task24\_1

17579 17583 TS 19 - 0 ./task24\_1

17584 17584 TS 19 - 0 sh -c ps -eLo pid,tid,cls,pri,rtprio,ni,cmd | grep ./task24\_1 > 24\_1res.txt

17586 17586 TS 19 - 0 grep ./task24\_1

(base) katya@katya:~/os/lb4$ gcc task24\_1.c -o task24\_1

(base) katya@katya:~/os/lb4$ sudo taskset 1 ./task24\_1

[sudo] пароль для katya:

SCHED\_FIFO

START: tid=18185

tid=18185

tid=18185

tid=18185

tid=18185

tid=18185

tid=18185

tid=18185

tid=18185

tid=18185

tid=18185

END: tid=18185

START: tid=18186

tid=18186

tid=18186

tid=18186

tid=18186

tid=18186

tid=18186

tid=18186

tid=18186

tid=18186

tid=18186

END: tid=18186

START: tid=18187

tid=18187

tid=18187

tid=18187

tid=18187

tid=18187

tid=18187

tid=18187

tid=18187

tid=18187

tid=18187

END: tid=18187

START: tid=18188

tid=18188

tid=18188

tid=18188

tid=18188

tid=18188

tid=18188

tid=18188

tid=18188

tid=18188

tid=18188

END: tid=18188

(base) katya@katya:~/os/lb4$ cat 24res.txt

cat: 24res.txt: Нет такого файла или каталога

(base) katya@katya:~/os/lb4$ cat 24\_1res.txt

18182 18182 TS 19 - 0 sudo taskset 1 ./task24\_1

18183 18183 TS 19 - 0 sudo taskset 1 ./task24\_1

18184 18184 FF 41 1 - ./task24\_1

18189 18189 FF 41 1 - sh -c ps -eLo pid,tid,cls,pri,rtprio,ni,cmd | grep ./task24\_1 > 24\_1res.txt

18193 18193 FF 41 1 - grep ./task24\_1

(base) katya@katya:~/os/lb4$ gcc task24\_1.c -o task24\_1

(base) katya@katya:~/os/lb4$ sudo taskset 1 ./task24\_1

SCHED\_RR

START: tid=18522

START: tid=18523

START: tid=18524

START: tid=18525

tid=18522

tid=18523

tid=18524

tid=18525

tid=18522

tid=18523

tid=18524

tid=18525

tid=18522

tid=18523

tid=18524

tid=18525

tid=18522

tid=18523

tid=18524

tid=18525

tid=18522

tid=18523

tid=18524

tid=18525

tid=18522

tid=18523

tid=18524

tid=18525

tid=18522

tid=18523

tid=18524

tid=18525

tid=18525

tid=18522

tid=18523

tid=18524

tid=18522

tid=18523

tid=18524

tid=18525

tid=18522

END: tid=18522

tid=18523

END: tid=18523

tid=18524

END: tid=18524

tid=18525

END: tid=18525

(base) katya@katya:~/os/lb4$ cat 24\_1res.txt

18519 18519 TS 19 - 0 sudo taskset 1 ./task24\_1

18520 18520 TS 19 - 0 sudo taskset 1 ./task24\_1

18521 18521 RR 41 1 - ./task24\_1

18521 18522 RR 41 1 - ./task24\_1

18521 18523 RR 41 1 - ./task24\_1

18521 18524 RR 41 1 - ./task24\_1

18521 18525 RR 41 1 - ./task24\_1

18526 18526 RR 41 1 - sh -c ps -eLo pid,tid,cls,pri,rtprio,ni,cmd | grep ./task24\_1 > 24\_1res.txt

18528 18528 RR 41 1 - grep ./task24\_1

Проведя эксперимент для всех трех политик, можно сказать, что наблюдается такая же конкуренция, как и лаб работе №3: при SCHED\_FIFO первый захвативший поток заканчивает свою работу полностью, а SCHED\_RR и SCHED\_OTHER чередуют потоки.

(base) katya@katya:~/os/lb4$ cat **task24\_2.c**

#include <signal.h>

#include <pthread.h>

#include <stdio.h>

#include <time.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/syscall.h>

pthread\_t t1, t2, t3, t4;

void work(){

int n = 0;

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 1000000000; j++)

n += 1;

printf("tid=%ld\n", syscall(SYS\_gettid));

}

}

// ТЕПЕРЬ ДЛЯ КАЖДОГО ПОТОКА УСТАНАВЛИВАЕМ СОБСТВЕННЫЙ ПРИОРИТЕТ

void\* thread1(void\*) {

pthread\_setschedprio(pthread\_self(), 10);

printf("START: tid=%ld, pri=%d\n", syscall(SYS\_gettid), 10);

work();

printf("END: tid=%ld\n", syscall(SYS\_gettid));

}

void\* thread2(void\*) {

pthread\_setschedprio(pthread\_self(), 20);

printf("START: tid=%ld, pri=%d\n", syscall(SYS\_gettid), 20);

work();

printf("END: tid=%ld\n", syscall(SYS\_gettid));

}

void\* thread3(void\*) {

pthread\_setschedprio(pthread\_self(), 30);

printf("START: tid=%ld, pri=%d\n", syscall(SYS\_gettid), 30);

work();

printf("END: tid=%ld\n", syscall(SYS\_gettid));

}

void\* thread4(void\*) {

pthread\_setschedprio(pthread\_self(), 40);

printf("START: tid=%ld, pri=%d\n", syscall(SYS\_gettid), 40);

work();

printf("END: tid=%ld\n", syscall(SYS\_gettid));

}

int main()

{

// меняем политику планирования родителя

struct sched\_param param;

param.sched\_priority = 90;

sched\_setscheduler(0, SCHED\_RR, &param);

switch (sched\_getscheduler(0))

{

case SCHED\_FIFO:

printf("SCHED\_FIFO\n");

break;

case SCHED\_RR:

printf("SCHED\_RR\n");

break;

case SCHED\_OTHER:

printf("SCHED\_OTHER\n");

break;

}

// запускаем потоки

pthread\_create(&t1, NULL, thread1, NULL);

pthread\_create(&t2, NULL, thread2, NULL);

pthread\_create(&t3, NULL, thread3, NULL);

pthread\_create(&t4, NULL, thread4, NULL);

system("ps -eLo pid,tid,cls,pri,rtprio,ni,cmd | grep ./task24\_2 > 24\_2res.txt");

pthread\_join(t1, NULL);

pthread\_join(t2, NULL);

pthread\_join(t3, NULL);

pthread\_join(t4, NULL);

return 0;

}(base) katya@katya:~/os/lb4$ gcc task24\_2.c -o task24\_2

(base) katya@katya:~/os/lb4$ sudo taskset 1 ./task24\_2

[sudo] пароль для katya:

SCHED\_RR

START: tid=19318, pri=40

tid=19318

tid=19318

tid=19318

tid=19318

tid=19318

tid=19318

tid=19318

tid=19318

tid=19318

tid=19318

END: tid=19318

START: tid=19317, pri=30

tid=19317

tid=19317

tid=19317

tid=19317

tid=19317

tid=19317

tid=19317

tid=19317

tid=19317

tid=19317

END: tid=19317

START: tid=19316, pri=20

tid=19316

tid=19316

tid=19316

tid=19316

tid=19316

tid=19316

tid=19316

tid=19316

tid=19316

tid=19316

END: tid=19316

START: tid=19315, pri=10

tid=19315

tid=19315

tid=19315

tid=19315

tid=19315

tid=19315

tid=19315

tid=19315

tid=19315

tid=19315

END: tid=19315

(base) katya@katya:~/os/lb4$ cat 24\_2res.txt

19312 19312 TS 19 - 0 sudo taskset 1 ./task24\_2

19313 19313 TS 19 - 0 sudo taskset 1 ./task24\_2

19314 19314 RR 130 90 - ./task24\_2

19314 19315 RR 50 10 - ./task24\_2

19314 19316 RR 60 20 - ./task24\_2

19314 19317 RR 70 30 - ./task24\_2

19314 19318 RR 80 40 - ./task24\_2

19319 19319 RR 130 90 - sh -c ps -eLo pid,tid,cls,pri,rtprio,ni,cmd | grep ./task24\_2 > 24\_2res.txt

19321 19321 RR 130 90 - grep ./task24\_2

Установив каждому потоку собственный приоритет с помощью pthread\_setschedprio и политику SCHED\_RR, можно заметить, что при более приоритетный поток забирает на себя все процессорное время (о чем и говорилась в предыдущей лаб работе). То есть сначала выполняется поток с приоритетом 40, затем 30, 20, 10.

Таким образом можно сказать, что потоки и процессы очень тесно связаны и имеют много общего.