|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

***к лабораторной работе №4***

***по дисциплине «Операционные системы»***

***по теме «Процессы. Системные вызовы fork() и exec().»***

|  |  |
| --- | --- |
| **Студент: Прохорова Любовь**  **Группа: ИУ7-53б**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель: Рязанова Н. Ю.** |  |

Москва.

2020 г.

Задание 1

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

int first\_child = fork();

if (first\_child == -1) {

perror("Can't fork");

exit(1);

}

else if (first\_child == 0) {

printf("Child: pid=%d, pidid=%d, groupid=%d**\n**", getpid(), getppid(), getpgrp());

sleep(3);

printf("Child: pid=%d, pidid=%d, groupid=%d**\n**", getpid(), getppid(), getpgrp());

return 0;

}

printf("Parent: pid=%d, childpid=%d, groupid=%d**\n**", getpid(), first\_child, getpgrp());

int second\_child = fork();

if (second\_child == -1) {

perror("Can't fork");

exit(1);

} else if (second\_child == 0) {

printf("Child: pid=%d, pidid=%d, groupid=%d**\n**", getpid(), getppid(), getpgrp());

sleep(3);

printf("Child: pid=%d, pidid=%d, groupid=%d**\n**", getpid(), getppid(), getpgrp());

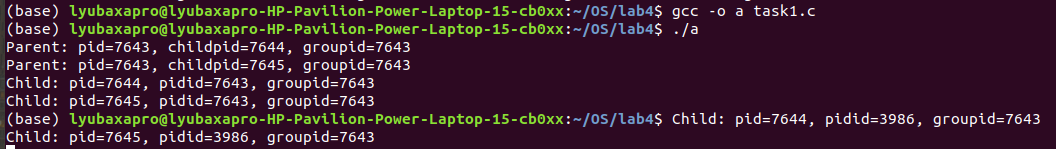
return 0;

}

printf("Parent: pid=%d, childpid=%d, groupid=%d**\n**", getpid(), second\_child, getpgrp());

return 0;

}



При завершении процесса-предка потомок, который продолжает выполняться, получает идентификатор предка (PPID), равный идентификатору процесса-посредника.



Задание 2

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main() {

int first\_child = fork();

int second\_child;

if ( first\_child == -1 ) {

perror("Couldn't fork.");

exit(1);

}

else if ( first\_child == 0 ) {

printf( "Child: pid=%d; group=%d; parent=%d**\n**", getpid(), getpgrp(), getppid() );

return 0;

}

printf( "Parent: pid=%d; group=%d; child=%d**\n**", getpid(), getpgrp(), first\_child );

second\_child = fork();

if ( second\_child == -1 ) {

perror("Couldn't fork.");

exit(1);

}

else if ( second\_child == 0 ) {

printf( "Child: pid=%d; group=%d; parent=%d**\n**", getpid(), getpgrp(), getppid() );

return 0;

}

printf( "Parent: pid=%d; group=%d; child=%d**\n**", getpid(), getpgrp(), second\_child );

if (first\_child != 0 && second\_child != 0) {

int status1;

int status2;

pid\_t ret\_value1 = wait(&status1);

if (WIFEXITED(status1))

printf("Parent: child %d finished with %d code.**\n**", ret\_value1, WEXITSTATUS(status1) );

else if (WIFSIGNALED(status1))

printf( "Parent: child %d finished from signal with %d code.**\n**", ret\_value1, WTERMSIG(status1));

else if (WIFSTOPPED(status1))

printf("Parent: child %d finished from signal with %d code.**\n**", ret\_value1, WSTOPSIG(status1));

pid\_t ret\_value2 = wait(&status2);

if (WIFEXITED(status2))

printf("Parent: child %d finished with %d code.**\n**", ret\_value2, WEXITSTATUS(status2) );

else if (WIFSIGNALED(status2))

printf( "Parent: child %d finished from signal with %d code.**\n**", ret\_value2, WTERMSIG(status2));

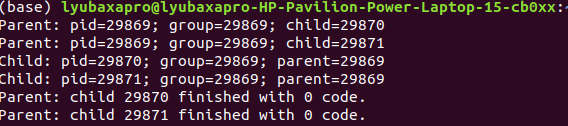
else if (WIFSTOPPED(status2))

printf("Parent: child %d finished from signal with %d code.**\n**", ret\_value2, WSTOPSIG(status2));

}

return 0;

}



Задание 3

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main() {

int first\_child = fork();

if ( first\_child == -1 ) {

perror("Couldn't fork.");

exit(1);

} else if ( first\_child == 0 ) {

printf( "Child: pid=%d; group=%d; parent=%d**\n\n**", getpid(), getpgrp(), getppid() );

if ( execl("/bin/ls", "ls", "-l", (char \*)NULL) == -1 ) {

perror( "Child couldn't exec." );

exit(1);

}

}

int second\_child = fork();

if ( second\_child == -1 ) {

exit(1);

} else if ( second\_child == 0 ) {

printf( "Child: pid=%d; group=%d; parent=%d**\n\n**", getpid(), getpgrp(), getppid() );

if ( execl("/bin/ps", "ps", "al", (char \*)NULL) == -1 ) {

perror( "Child couldn't exec." );

exit(1);

}

}

if (first\_child != 0 && second\_child != 0) {

printf( "Parent: pid=%d; group=%d; first\_child=%d, second\_child=%d**\n**", getpid(), getpgrp(), first\_child, second\_child );

int status1, status2;

pid\_t ret\_value1 = wait( &status1 );

if ( WIFEXITED(status1) )

printf("Parent: child %d finished with %d code.**\n\n**", ret\_value1, WEXITSTATUS(status1) );

else if ( WIFSIGNALED(status1) )

printf( "Parent: child %d finished from signal with %d code.**\n\n**", ret\_value1, WTERMSIG(status1));

else if ( WIFSTOPPED(status1) )

printf("Parent: child %d finished from signal with %d code.**\n\n**", ret\_value1, WSTOPSIG(status1));

pid\_t ret\_value2 = wait( &status2 );

if ( WIFEXITED(status2) )

printf("Parent: child %d finished with %d code.**\n\n**", ret\_value2, WEXITSTATUS(status2) );

else if ( WIFSIGNALED(status2) )

printf( "Parent: child %d finished from signal with %d code.**\n\n**", ret\_value2, WTERMSIG(status2));

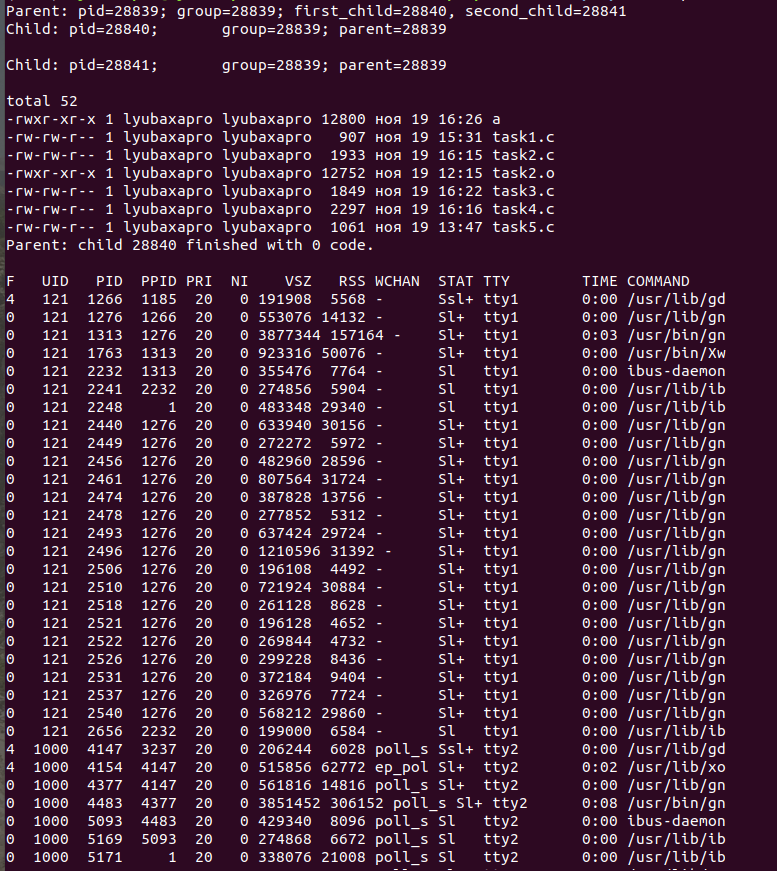
else if ( WIFSTOPPED(status2) )

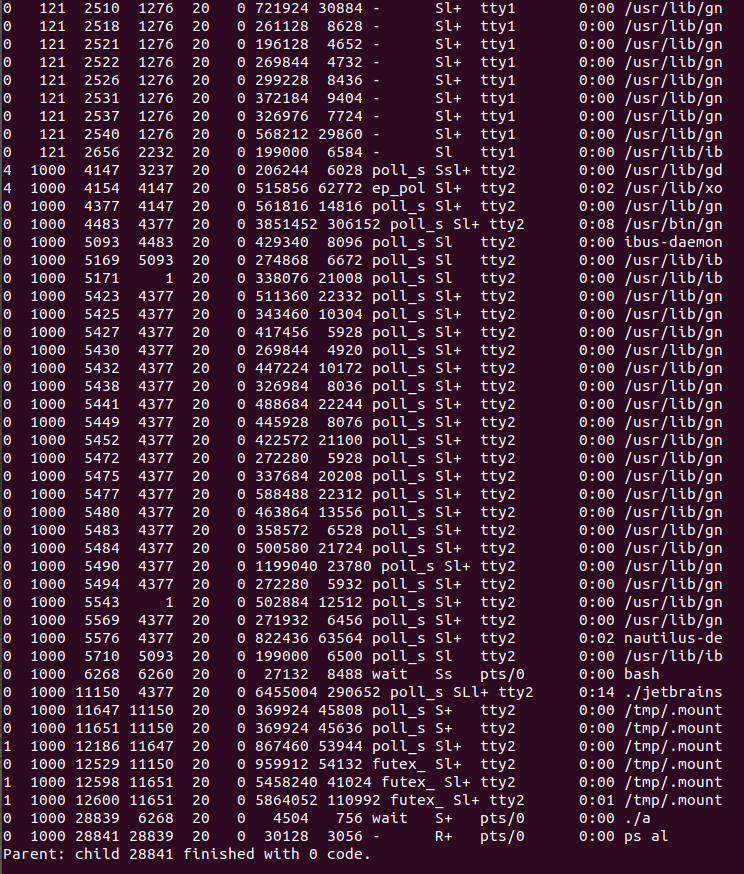
printf("Parent: child %d finished from signal with %d code.**\n\n**", ret\_value2, WSTOPSIG(status2));

}

return 0;

}





Задание 4

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <wait.h>

int main() {

int descr[2];

if (pipe(descr) == -1) {

perror( "Couldn't pipe." );

exit(1);

}

int first\_child = fork();

if (first\_child == -1) {

perror("Couldn't fork.");

exit(1);

} else if (first\_child == 0) {

close( descr[0] );

char msg1[] = "Message from the first child.";

write(descr[1], msg1, 64);

exit(0);

}

int second\_child = fork();

if (second\_child == -1) {

perror( "Couldn't fork." );

exit(1);

} else if (second\_child == 0) {

close( descr[0] );

char msg2[] = "Message from the second child.";

write(descr[1], msg2, 64);

exit(0);

}

if (first\_child != 0 && second\_child != 0) {

close(descr[1]);

char msg1[64];

read(descr[0], msg1, 64);

char msg2[64];

read(descr[0], msg2, 64);

printf("Parent: reads %s %s**\n**", msg1, msg2);

int status1, status2;

pid\_t ret\_value1 = wait(&status1);

if (WIFEXITED(status1))

printf("Parent: child %d finished with %d code.**\n\n**", ret\_value1, WEXITSTATUS(status1));

else if (WIFSIGNALED(status1))

printf("Parent: child %d finished from signal with %d code.**\n\n**", ret\_value1, WTERMSIG(status1));

else if (WIFSTOPPED(status1))

printf("Parent: child %d finished from signal with %d code.**\n\n**", ret\_value1, WSTOPSIG(status1));

pid\_t ret\_value2 = wait(&status2);

if (WIFEXITED(status2))

printf("Parent: child %d finished with %d code.**\n\n**", ret\_value2, WEXITSTATUS(status2));

else if (WIFSIGNALED(status2))

printf( "Parent: child %d finished from signal with %d code.**\n\n**", ret\_value2, WTERMSIG(status2));

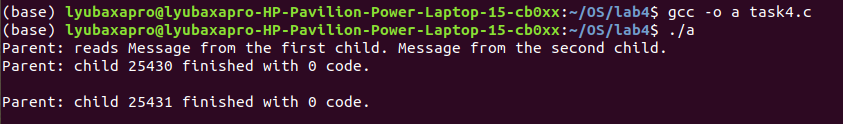
else if (WIFSTOPPED(status2))

printf("Parent: child %d finished from signal with %d code.**\n\n**", ret\_value2, WSTOPSIG(status2));

}

return 0;

}



Задание 5

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <wait.h>

#include <signal.h>

#include <time.h>

int flag = 0;

void catcher(int signum) {

printf("Proccess catched signal #%d**\n**", signum);

flag = 1;

}

int main() {

int descr[2];

if (pipe(descr) == -1) {

perror( "Couldn't pipe." );

exit(1);

}

signal(SIGINT, catcher);

int first\_child = fork();

if (first\_child == -1) {

perror("Couldn't fork.");

exit(1);

}

else if (first\_child == 0) {

sleep(3);

char msg1[64];

if (flag)

sprintf(msg1, "Message from child 1 - signal was catched.");

else

sprintf(msg1, "Message from child 1 - signal NOT catched.");

close(descr[0]);

(write(descr[1], msg1, 64));

exit(0);

}

int second\_child = fork();

if (second\_child == -1) {

perror( "Couldn't fork." );

exit(1);

}

else if (second\_child == 0) {

sleep(3);

char msg2[64];

if (flag)

sprintf(msg2, "Message from child 2 - signal was catched.");

else

sprintf(msg2, "Message from child 2 - signal NOT catched.");

close(descr[0]);

(write(descr[1], msg2, 64));

exit(0);

}

if (first\_child != 0 && second\_child != 0) {

int status1, status2;

pid\_t ret\_value1 = wait(&status1);

if (WIFEXITED(status1))

printf("Parent: child %d finished with %d code.**\n\n**", ret\_value1, WEXITSTATUS(status1));

else if (WIFSIGNALED(status1))

printf("Parent: child %d finished from signal with %d code.**\n\n**", ret\_value1, WTERMSIG(status1));

else if (WIFSTOPPED(status1))

printf("Parent: child %d finished from signal with %d code.**\n\n**", ret\_value1, WSTOPSIG(status1));

pid\_t ret\_value2 = wait(&status2);

if (WIFEXITED(status2))

printf("Parent: child %d finished with %d code.**\n\n**", ret\_value2, WEXITSTATUS(status2));

else if (WIFSIGNALED(status2))

printf( "Parent: child %d finished from signal with %d code.**\n\n**", ret\_value2, WTERMSIG(status2));

else if (WIFSTOPPED(status2))

printf("Parent: child %d finished from signal with %d code.**\n\n**", ret\_value2, WSTOPSIG(status2));

close(descr[1]);

char msg1[64];

read(descr[0], msg1, 64);

char msg2[64];

read(descr[0], msg2, 64);

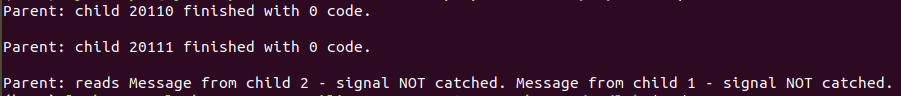
printf("Parent: reads %s %s**\n**", msg1, msg2);

return 0;

}

}

Вывод программы, если сигнал не был получен.



Вывод программы, если сигнал был получен.

