Q1

// Receiver

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

int main()

{

int a = 0;

int msgid = msgget((key\_t)1234, 0666 | IPC\_CREAT);

if (msgid == -1)

{

fprintf(stderr, "Getting message failed\n");

exit(EXIT\_FAILURE);

}

if (msgrcv(msgid, &a, sizeof(int), 0, 0) == -1)

{

printf("Sending message failed\n");

exit(EXIT\_FAILURE);

}

else

{

printf("Number recieved is %d\n", a);

int p = a;

int r = 0, s = 0;

while (p != 0)

{

r = p % 10;

s = (s \* 10) + r;

p = p / 10;

}

if (s == a)

printf("Palindrome\n");

else

printf("Not a Palindrome \n");

}

exit(EXIT\_SUCCESS);

}

// Sender

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

int main()

{

int a = 0;

int msgid = msgget((key\_t)1234, 0666 | IPC\_CREAT);

if (msgid == -1)

{

fprintf(stderr, "Getting message failed\n");

exit(1);

}

printf("Enter a number");

scanf("%d", &a);

if (msgsnd(msgid, &a, sizeof(int), 0) == -1)

{

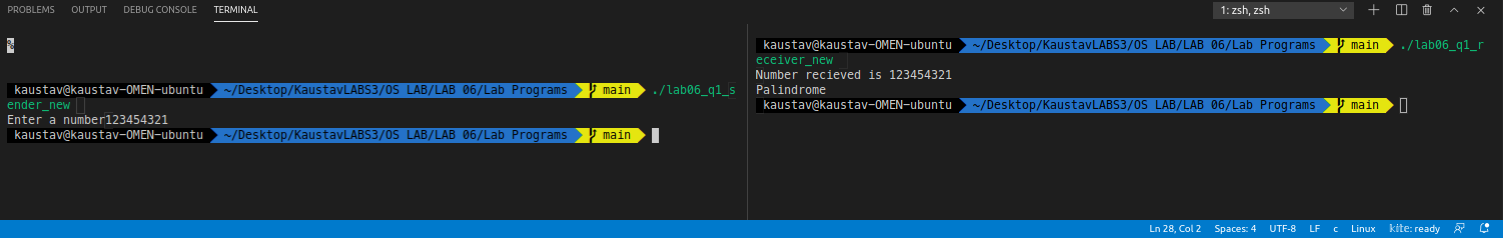
printf("Sending message failed\n");

exit(EXIT\_FAILURE);

}

exit(0);

}



Q2

// Parent

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

struct shared\_use

{

char c;

char next\_char;

};

int main()

{

int running = 1;

void \*shared\_memory = (void \*)0;

int shmid;

struct shared\_use \*stuff;

srand((unsigned int)getpid());

shmid = shmget((key\_t)1234, sizeof(struct shared\_use), 0666 | IPC\_CREAT);

if (shmid == -1)

{

fprintf(stderr, "shmget failure!!\n");

exit(EXIT\_FAILURE);

}

shared\_memory = shmat(shmid, (void \*)0, 0);

if (shared\_memory == (void \*)-1)

{

fprintf(stderr, "shmat failure\n");

exit(EXIT\_FAILURE);

}

stuff = (struct shared\_use \*)shared\_memory;

char ch;

printf("Enter a character:\n");

scanf("%c", &ch);

stuff->c = ch;

printf("current char: %c\n", stuff->c);

printf("Waiting for child process to change\n");

sleep(4);

printf("Current character: %c\n", stuff->c);

if (shmdt(shared\_memory) == -1)

{

fprintf(stderr, "shmdt failure\n");

exit(EXIT\_FAILURE);

}

}

// Child

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

struct shared\_use

{

char c;

char next\_char;

};

int main()

{

int running = 1;

void \*shared\_memory = (void \*)0;

int shmid;

struct shared\_use \*stuff;

char buffer;

shmid = shmget((key\_t)1234, sizeof(struct shared\_use), 0666 | IPC\_CREAT);

if (shmid == -1)

{

fprintf(stderr, "shmget failure\n");

exit(1);

}

shared\_memory = shmat(shmid, (void \*)0, 0);

if (shared\_memory == (void \*)-1)

{

fprintf(stderr, "shmat failure\n");

exit(1);

}

stuff = (struct shared\_use \*)shared\_memory;

printf("Current character: %c\n", stuff->c);

stuff->c++;

if (shmdt(shared\_memory) == -1)

{

fprintf(stderr, "shmdt failure\n");

exit(1);

}

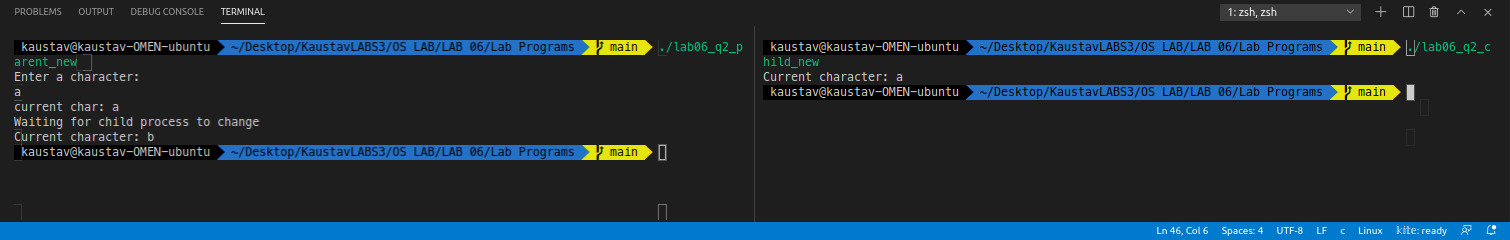
if (shmctl(shmid, IPC\_RMID, 0) == -1)

{

fprintf(stderr, "shmctl(IP\_RMID) failure\n");

exit(1);

}

}