**Thread 1**

#include <stdio.h>

#include <pthread.h>

#include<unistd.h>

void\* threadFunction(void\* args)

{

int i;

for(i=1;i<=5;++i)

{

printf("I am new thread.\n");

sleep(1);

}

}

int main()

{

pthread\_t id;

int res,i;

res=pthread\_create(&id,NULL,&threadFunction,NULL);

if(res==0)

{

printf("Thread created successfully.\n");

}

else

{

printf("Thread not created.\n");

return 0;

}

for(i=1;i<=2;++i)

{

printf("I am main thread.\n");

sleep(1);

}

return 0;

}

To compile:

$ cc thr1.c -o thr.out –lpthread

$./thr.out

**Thread 2**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include <pthread.h>

void \*thread\_function(void \*arg);

char message[] = "Hello World";

int main()

{

int res;

pthread\_t a\_thread;

void \*thread\_result;

res = pthread\_create(&a\_thread, NULL, thread\_function, message);

if (res != 0)

{

perror("Thread creation failed");

exit(EXIT\_FAILURE);

}

printf("Waiting for thread to finish...\n");

res = pthread\_join(a\_thread, &thread\_result);

if (res != 0)

{

perror("Thread join failed");

exit(EXIT\_FAILURE);

}

printf("Thread joined, it returned: %s\n", (char \*)thread\_result);

printf("Message is now: %s\n", message);

exit(EXIT\_SUCCESS);

}

void \*thread\_function(void \*arg)

{

printf("Thread\_function is running. Argument was: %s\n", (char \*)arg);

sleep(3);

strcpy(message, "Bye!");

pthread\_exit("Thank you for the CPU time");

}

**To compile:**

$ cc thr1.c -o thr.out –lpthread

$./thr.out

**Pipes 1**

------------------------------------------

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

//Writing to external command

int main()

{

FILE \*write\_fp;

char buffer[BUFSIZ + 1];

sprintf(buffer, "The quick brown fox is sleeping.\n");

write\_fp = popen("cat > newfile.txt", "w");

if (write\_fp != NULL)

{

fwrite(buffer, sizeof(char), strlen(buffer), write\_fp);

pclose(write\_fp);

exit(EXIT\_SUCCESS);

}

exit(EXIT\_FAILURE);

}

**Pipes 2**

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

//Communication between child and parent

int main()

{

int data\_processed;

int file\_pipes[2];

const char some\_data[] = "Hello World";

char buffer[BUFSIZ + 1];

pid\_t fork\_result;

memset(buffer, '\0', sizeof(buffer));

if (pipe(file\_pipes) == 0)

{

fork\_result = fork();

if (fork\_result == -1)

{

fprintf(stderr, "Fork failure");

exit(EXIT\_FAILURE);

}

else if (fork\_result == 0) //Child

{

sleep(2);

data\_processed = read(file\_pipes[0], buffer, BUFSIZ);

printf("Read %d bytes: %s\n", data\_processed, buffer);

exit(EXIT\_SUCCESS);

}

else //Parent

{

data\_processed = write(file\_pipes[1], some\_data,

strlen(some\_data));

printf("Wrote %d bytes\n", data\_processed);

}

}

exit(EXIT\_SUCCESS);

}