```
#include <stdio.h>
#include <unistd.h>
#include<string.h>
#include <stdbool.h>
void syserr(char str[])
{
        perror(str);
        exit(1);
} /* sys_arr */

int main(int argc, char *argv[])
{
    int pfd[2],stat,pid,i=0;
    char msg[80],temp[80];
    char msg2[80],msg3[80];
    char *result2=NULL;
    bool flag=false;
    if (pipe(pfd) == -1)
        syserr("pipe");
    printf("Welcome in my extended mini shell. Type 'exit' to terminate.\n");
    printf("minishell2>");
    gets(msg);
    //printf("asd:%s\n",msg);
    while( strcmp( msg, "exit" ) != 0)
    {

        char Pipe[] = "|";
            char *result = NULL;
        char *argv2[3];
            strcpy(temp,msg);
        result = strtok(msg, Pipe );
        if(strcmp (temp, result) !=0)
        {
            strcpy(msg2,result);
            flag=true;
            result = strtok( NULL, Pipe );
            strcpy(msg3,result);
        }
        if(flag)
        {
            switch(fork())
            {
                case -1:
                    syserr("fork");
                case 0:
                    if(argc>1 && strcmp(argv[1], "-debug" ) == 0)
                    {
                        printf("INFO: Pipe detected. Command 1: '%s' and Command 2: '%s'\n",msg2
                        ,msg3);
                        printf("INFO: Making pipe\n");
                        printf("INFO: Child started PID[%d] command '%s'\n ",getppid(),msg2);
                    }
                    if (dup2(pfd[1], 1) == -1)
                    syserr("dup2");
                    if (close(pfd[0]) == -1 || close(pfd[1]) == -1)
                        syserr("close");

                    result2 = strtok(msg," ");
                                int i;
                                i=0;
                                while( result2 != NULL )
                                {
                                    argv2[i]=result2;
                                     i++;
                                     result2 = strtok(NULL, "" );
                                }

                                 argv2[i]=NULL;
                                    execvp(argv2[0], argv2);

                    //execlp(msg2, msg2, NULL);
                    pid=getppid();
                    syserr("execlp");
            }
            switch(fork())
            {
                case -1:
                    syserr("fork");
                case 0:
                    if(argc>1 && strcmp(argv[1], "-debug" ) == 0)
                    {
                        printf("INFO: Child started PID[%d] command '%s'\n ",getppid(),msg3);
                    }
                    if (dup2(pfd[0], 0) == -1)
                    syserr("dup2");
                    if (close(pfd[0]) == -1 || close(pfd[1]) == -1)
                    syserr("close");
                    execlp(msg3, msg3, NULL);
                    printf("INFO: Child with PID[%d]terminated, continue waiting commands\n",
                    getppid());
                    printf("INFO: Child with PID[%d]terminated, continue waiting commands\n",pid
                    );
                    syserr("execlp2");
            }
            if (close(pfd[0]) == -1 || close(pfd[1]) == -1)
                        syserr("close");
            flag=false;
        }
        else
        {
            switch (fork())
                        {
                          case -1: // error
                                perror("fork call");
                                exit(1);
                          case 0:
                if(argc>1 && strcmp(argv[1], "-debug" ) == 0)
                {
                    printf("INFO: No pipe detected, creating child for command '%s'\n", msg);
                    printf("INFO: Child started PID[%d] command '%s'\n",getppid(),msg);
                                }

                    result2 = strtok(msg," ");
            i=0;
                    while( result2 != NULL )
                    {
                argv2[i]=result2;
                i++;
                             result2 = strtok(NULL, "" );
                    }
            argv2[i]=NULL;
                execvp(argv2[0], argv2);

                        if(argc>1 && strcmp(argv[1], "-debug" ) == 0)
                            printf("INFO: Child with PID[%d]terminated, continue waiting
                            commands\n",getppid());
            exit(3);


            }

        }
        //while(wait(NULL)>0);
        wait(&stat);
        wait(&stat);
          if (close(pfd[0]) == -1 || close(pfd[1]) == -1);
          if (pipe(pfd) == -1);

        printf("minishell2>");


        gets(msg);

    }/*while*/
    return 0;
} /* main */
```