**Atoi**  
#include <stdio.h>

int ft\_atoi(const char \*str)

{

int res = 0;

int neg = 1;

while(\*str && ( \*str == ' ' || \*str == '\t' || \*str == '\f' || \*str == '\v' || \*str == '\r' || \*str == '\n'))

str++;

if(\*str == '+' || \*str == '-')

{

if(\*str == '-')

neg = -1;

str++;

}

while(\*str <= '9' && \*str >= '0')

{

if(res != ((res << 3) + (res << 1) + (\*str - '0')) / 10)

return (neg == 1 ? -1 : 0 );

res = (res << 3) + (res << 1) + \*str++ - '0';

}

return ((int) res \* neg);

}

int main (int argc, char \*\*argv)

{

if (argc == 2)

printf("%d\n", ft\_atoi(argv[1]));

return (0);

}

Alpha Mirror

#include <unistd.h>

void alpha\_mirror (char \*str)

{

while(\*str)

{

if(\*str >= 'a' && \*str <= 'z')

\*str = 'm' - (\*str - 'n');

else if(\*str >= 'a' && \*str <= 'Z')

\*str = 'M' - (\*str - 'N');

write(1, &(\*str), 1);

str++;

}

write(1, "\n", 1);

}

int main(int argc, char \*\*argv)

{

if(argc == 2)

alpha\_mirror(argv[1]);

return 0;

}

Inter

#include <unistd.h>

#include <stdio.h>

void inter (char \*str1, char \*str2)

{

char ascii[256] = {0};

while(\*str2)

ascii[((int)\*str2++)] = 1;

while(\*str1)

{

if(ascii[((int)\*str1)] == 1)

{

write(1, &(\*str1), 1);

ascii[((int)\*str1)] = 0;

}

str1++;

}

}

int main (int argc, char \*\*argv)

{

if(argc == 3)

inter(argv[1],argv[2]);

write(1, "\n", 1);

return (0);

}

Last Word

#include <unistd.h>

#include <stdio.h>

void last\_word(char \*str)

{

char \*beg;

int a = 0;

while(\*str && (\*str == ' ' || \*str == '\t'))

str++;

beg = str;

while(\*str)

{

if(\*str == ' ' || \*str == '\t')

a = 1;

if(a == 1 && \*str != ' ' && \*str != '\t')

{

a = 0;

beg = str;

}

str++;

}

while(\*beg && \*beg != ' ' && \*beg != '\t')

{

write(1, &(\*beg), 1);

beg++;

}

}

int main (int argc, char \*\*argv)

{

if(argc == 2)

last\_word(argv[1]);

write(1, "\n", 1);

return (0);

}

Max

#include <stdio.h>

int max (int \*tab, unsigned int len)

{

unsigned int i = -1;

int max;

if(!tab)

return 0;

if(len == 0)

return 0;

max = tab[0];

while(++i < len)

if(max < tab[i])

max = tab[i];

return(max);

}

int main()

{

int tab[10] = {};

printf("%d\n",max(tab,10));

return 0;

}

Reverse Bits

#include <unistd.h>

#include <stdio.h>

unsigned char reverse\_bits(unsigned char octet)

{

unsigned char print = 1;

unsigned char res = 0;

while(print < 128)

{

if(print & octet)

res += 1;

res <<= 1;

print <<= 1;

}

return (res);

}

int main(int argc, char \*\*argv)

{

char c = reverse\_bits(2);

printf("%d\n", c);

return (0);

Str Dup

#include <stdio.h>

#include <stdlib.h>

char \*ft\_strdup(char \*src)

{

int len = 0;

char \*str;

if(!src)

return (NULL);

while (src[len++]);

str = (char\*)malloc(sizeof(char) \* (len + 1));

str[len + 1] = '\0';

while (len-- > 0)

str[len-1] = src[len-1];

return (str);

}

int main (int argc, char \*\*argv)

{

if(argc == 2)

printf("%s\n", ft\_strdup(argv[1]));

return (0);

}

Union

#include <stdio.h>

#include <unistd.h>

void my\_union (char \*\*str)

{

char ascii[256] = {0};

int j = 0;

while(str[1][j])

{

if(ascii[((int)str[1][j])] == 0)

{

write(1, &(str[1][j]), 1);

ascii[((int)str[1][j])] = 1;

}

j++;

}

j = 0;

while(str[2][j])

{

if(ascii[(int)str[2][j]] == 0)

{

write(1, &(str[2][j]), 1);

ascii[((int)str[2][j])] = 1;

}

j++;

}

}

int main(int argc, char \*\*argv)

{

if(argc == 3)

my\_union(argv);

return 0;

}

Wbmathc

#include <stdio.h>

#include <unistd.h>

void wdmatch (char \*\*str)

{

char \*tmp = str[1];

while(\*str[1] && \*str[2])

{

if(\*str[1] == \*str[2])

str[1]++;

str[2]++;

}

if(!(\*str[1]))

while(\*tmp)

write(1, tmp++, 1);

}

int main(int argc, char \*\*argv)

{

if(argc == 3)

wdmatch(argv);

write(1, "\n", 1);

return 0;

}

Do op

#include <stdio.h>

#include <stdlib.h>

Int main(int argc, char \*\*argv)

{

if (argc == 4)

{

if (\*argv[2] == '+')

printf("%d", atoi(argv[1]) + atoi(argv[3]));

else if (\*argv[2] == '-')

printf("%d", atoi(argv[1]) - atoi(argv[3]));

else if (\*argv[2] == '\*')

printf("%d", atoi(argv[1]) \* atoi(argv[3]));

else if (\*argv[2] == '/')

printf("%d", atoi(argv[1]) / atoi(argv[3]));

else if (\*argv[2] == '%')

printf("%d", atoi(argv[1]) % atoi(argv[3]));

}

printf("\n");

return (0);

}

Is power of 2

Int is\_power\_of\_2 (unsigned int n)

{

Return ((n z (-n)) == n ? 1 : 0);

}\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print Bits

#include <unistd.h>

void print\_bits(unsigned char octet)

{

int i;

i = 128;

while (octet >= 0 && i)

{

(octet / i) ? write(1, "1", 1) : write(1, "0", 1);

(octet / i) ? octet -= i : 0;

i /= 2;

}

}

Add prime sum

#include <unistd.h>

void my\_print\_char(char c)

{

write(1, &c, 1);

}

void my\_prinbr(unsigned long long nb)

{

if(nb >= 10)

my\_prinbr(nb / 10);

my\_print\_char(nb % 10 + 48);

}

int my\_atoi(char \*nb)

{

int sum = 0;

while(\*nb == '\n' || \*nb == '\t' || \*nb == '\v' || \*nb == '\f' || \*nb == ' ')

nb++;

if(\*nb == '-')

return (0);

if(\*nb == '+')

nb++;

while(\*nb <= '9' && \*nb >= '0')

{

if(sum != ((sum << 3) + (sum << 1) + \*nb - '0') / 10)

return(1);

sum = ((sum << 3) + (sum << 1) + \*nb++ - '0');

}

return(sum);

}

int is\_prime(int nb)

{

int i = 2;

if(nb == 2)

return (nb);

while(i \* i <= nb)

{

if(nb % i == 0)

return (0);

i++;

}

return (nb);

}

unsigned long long add\_prime\_sum(int nb)

{

unsigned long long sum = 0;

while(nb > 1)

{

sum += is\_prime(nb);

nb--;

}

return(sum);

}

int main(int argc, char \*\*argv)

{

int matoi;

unsigned long long res;

if(argc == 2)

{

if((matoi = my\_atoi(argv[1])) == 0)

return (1);

res = add\_prime\_sum(matoi);

my\_prinbr(res);

}

write(1, "\n", 1);

return (0);

}

Epur str

#include <unistd.h>

void epur\_str (char \*str)

{

int a = 1;

while(\*str)

{

if(\*str == ' ' && a)

{

write(1, str, 1);

a = 0;

}

if(\*str != ' ')

{

a = 1;

write(1, str, 1);

}

str++;

}

}

int main(int argc, char \*\*argv)

{

if(argc == 2)

epur\_str(argv[1]);

write(1, "\n", 1);

return(0);

}

Expand str

#include <unistd.h>

#include <stdio.h>

void expand\_str(char \*str)

{

int count;

int flag = 0;

char \* tmp = str;

while(\*tmp)

tmp++;

while(\*tmp == ' ' || \*tmp == '\t')

tmp--;

while(str < tmp)

{

count = 3;

while((\*str == ' ' || \*str == '\t') && str <tmp)

str++;

while(\*str != ' ' && \*str != '\t' && str < tmp)

{

flag = 1;

write(1, str++, 1);

}

while(count > 0 && flag == 1 && str < tmp - 1)

{

write(1, " ", 1);

if(count-- == 0)

flag = 0;

}

}

}

Ft\_atoi\_base

int isblank(char c)

{

if (c <= 32)

return (1);

return (0);

}

int isvalid(char c, int base)

{

char digits[17] = "0123456789abcdef";

char digits2[17] = "0123456789ABCDEF";

while (base--)

if (digits[base] == c || digits2[base] == c)

return (1);

return (0);

}

int value\_of(char c)

{

if (c >= '0' && c <= '9')

return (c - '0');

else if (c >= 'a' && c <= 'f')

return (c - 'a' + 10);

else if (c >= 'A' && c <= 'F')

return (c - 'A' + 10);

return (0);

}

int ft\_atoi\_base(char \*str, int str\_base)

{

int result;

int sign;

result = 0;

while (isblank(\*str))

str++;

sign = (\*str == '-') ? -1 : 1;

(\*str == '-' || \*str == '+') ? ++str : 0;

while (isvalid(\*str, str\_base))

result = result \* str\_base + value\_of(\*str++);

return (result \* sign);

}

List size

#include "ft\_list.h"

#include <stdio.h>

#include <stdlib.h>

int ft\_list\_size(t\_list \*begin\_list)

{

int len = 0;

if(!begin\_list)

return (0);

while(begin\_list)

{

len++;

begin\_list = begin\_list->next;

}

return (len);

}

int main()

{

t\_list \*t1 = (t\_list\*)malloc(sizeof(t\_list));

t\_list \*t2 = (t\_list\*)malloc(sizeof(t\_list));

t\_list \*t3 = (t\_list\*)malloc(sizeof(t\_list));

t\_list \*t4 = (t\_list\*)malloc(sizeof(t\_list));

t\_list \*t5 = (t\_list\*)malloc(sizeof(t\_list));

t\_list \*t6 = (t\_list\*)malloc(sizeof(t\_list));

t6->next = NULL;

t6->data = "6";

t5->data = "5";

t5->next = t6;

t4->data = "4";

t4->next = t5;

t3->data = "3";

t3->next = t4;

t2->data = "2";

t2->next = t3;

t1->data = "1";

t1->next = t2;

printf("%d\n", ft\_list\_size(t1));

return 0;

}

typedef struct s\_list

{

struct s\_list \*next;

void \*data;

} t\_list;

Ft\_range

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

int \*ft\_rrange(int start, int end)

{

int len = (end - start) > 0 ? end - start +1 : start - end +1;

int \* res = (int\*)malloc(sizeof(int) \* len);

while(len--)

{

if(start <= end)

res[len] = start++;

else

res[len] = start--;

}

return (res);

}

int main()

{

int end = -3;

int start = 0;

int len = (end - start) > 0 ? end - start +1 : start - end +1;

int \*a = ft\_rrange(start,end);

int i = 0;

while(i < len)

printf("%d\n", a[i++]);

return 0;

}

Hide np

#include <unistd.h>

int hidenp (char \*\*str)

{

while(\*str[1] && \*str[2])

{

if(\*str[1] == \*str[2])

str[1]++;

str[2]++;

}

if(\*str[1])

return (0);

return(1);

}

int main(int argc, char \*\*argv)

{

char c;

if(argc == 3)

{

c = hidenp(argv) + 48;

write(1, &c, 1);

}

write(1, "\n", 1);

return (0);

}

Param sum

#include <unistd.h>

void ft\_putchar(char c)

{

write(1, &c, 1);

}

void ft\_putnbr(int n)

{

if (n > 9)

ft\_putnbr(n / 10);

ft\_putchar(n % 10 +'0');

}

int main(int ac, char \*\*av)

{

av = 0;

ft\_putnbr(ac - 1);

write(1, "\n", 1);

}

Pgcd

#include <stdio.h>

#include <stdlib.h>

int pgcd(int a, int b)

{

int c = (a > b) ? b : a;

while(c > 0)

{

if(!(a % c) && !(b % c))

return(c);

c--;

}

return (1);

}

int main(int argc, char \*\*argv)

{

if(argc == 3)

{

printf("%d", pgcd(atoi(argv[1]), atoi(argv[2])));

}

printf("\n");

return 0;

}

Print hex

#include <unistd.h>

int my\_atoi(char \*s)

{

int res = 0;

while(\*s == ' ' || \*s == '\t' || \*s == '\f' || \*s == '\v' || \*s == '\r' || \*s == '\r')

s++;

if(\*s == '-')

return (0);

if(\*s == '+')

s++;

while(\*s <= '9' && \*s >= '0')

{

if(res != ((res << 3) + (res << 1) + \*s - '0') / 10)

return (0);

res = (res << 3) + (res << 1) + \*s++ - '0';

}

return (res);

}

void print\_hex (int n)

{

if(n == 0)

return ;

if (n >= 16)

print\_hex(n / 16);

n %= 16;

n += (n < 10) ? '0' : 'a' - 10;

write(1, &n, 1);

}

int main(int argc, char \*\*argv)

{

if(argc == 2)

print\_hex(my\_atoi(argv[1]));

write(1, "\n", 1);

return 0;

}

Rstr\_capitalizer

#include <stdio.h>

char \*ft\_strcapitalize(char \*str)

{

int i;

char c;

int space;

c = ' ';

i = 0;

while (str[i])

i++;

while (--i >= -1)

{

space = 1;

if (i + 1 != '\0')

c = str[i + 1];

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9'))

space = 0;

if (space == 1 && str[i] >= 'a' && str[i] <= 'z')

str[i] = str[i] - 32;

if (space == 0 && str[i] >= 'A' && str[i] <= 'Z')

str[i] = str[i] + 32;

}

return (str);

}

int main(int argc, char \*\*argv)

{

if(argc == 2)

printf("%s\n", ft\_strcapitalize(argv[1]));

return 0;

}

Tab mult

#include <unistd.h>

void ft\_putchar(char c)

{

write(1, &c, 1);

}

void ft\_putposnbr(int n)

{

if (n > 9)

ft\_putposnbr(n / 10);

ft\_putchar(n % 10 + '0');

}

int ft\_atoi\_osefvalid(char \*s)

{

int r;

r = 0;

while (\*s >= '0' && \*s <= '9')

r = r \* 10 + \*s++ - '0';

return (r);

}

void tab\_mult(int n)

{

int i = 1;

while (i < 10)

{

ft\_putposnbr(i);

write(1," x ", 3);

ft\_putposnbr(n);

write(1, " = ", 3);

ft\_putposnbr(i++ \* n);

write(1, "\n", 1);

}

}

int main(int ac, char \*\*av)

{

if (ac == 2)

tab\_mult(ft\_atoi\_osefvalid(av[1]));

else

write(1, "\n", 1);

return (0);

}

Ft ragne

#include <stdlib.h>

int \*ft\_rrange(int start, int end)

{

int \*r;

int len;

len = (end >= start) ? end - start + 1 : start - end + 1;

if (!(r = (int\*)malloc(sizeof(int) \* len)))

return (NULL);

while (len--)

r[len] = (end >= start) ? start++ : start--;

return (r);

}

Aff\_a

#include <unistd.h>

int main(int argc, char \*\*argv)

{

if (argc != 2)

write(1, "a", 1);

else

{

while (\*argv[1])

{

if (\*argv[1] == 'a')

{

write(1, "a", 1);

break;

}

argv[1]++;

}

}

write(1, "\n", 1);

return (0);

}

First param

#include <unistd.h>

int main(int ac, char \*\*av)

{

if (ac > 1)

while (\*av[1])

write(1, av[1]++, 1);

write(1, "\n", 1);

return (0);

}

Last param

#include <unistd.h>

int main(int argc, char \*\*argv)

{

if (argc > 1)

while (\*argv[argc - 1])

write(1, argv[argc - 1]++, 1);

write(1, "\n", 1);

}

Search and replace

#include <unistd.h>

void ft\_putchar(char c)

{

write(1, &c, 1);

}

void sar(char \*str, char a, char b)

{

int i;

i = 0;

while(str[i])

{

if ( str[i] == a)

{

ft\_putchar(b);

}

else

{

ft\_putchar(str[i]);

}

i++;

}

}

int main(int argc, char \*\*argv)

{

if(argc == 4 && argv[2][1] == '\0' && argv[3][1] == '\0')

{

sar(argv[1], argv[2][0], argv[3][0]);

}

ft\_putchar('\n');

return (0);

}

Str cpy

char \*ft\_strcpy(char \*s1, char \*s2)

{

int i;

i = 0;

while (s2[i])

{

s1[i] = s2[i];

i++;

}

s1[i] = s2[i];

return (s1)

}

Strlen

int ft\_strlen(char \*str)

{

int i;

i = 0;

while (str[i])

i++;

return (i);

}

Putstr

#include <unistd.h>

void ft\_putstr(char \*str)

{

while (\*str)

write (1, str++, 1);

}

Rev print

#include <unistd.h>

int main(int argc, char \*\*argv)

{

int len;

len = 0;

if (argc == 2)

{

while (argv[1][len])

len++;

while (len > 0)

write (1, &(argv[1][--len]), 1);

return (0);

}

write (1, "\n", 1);

}

Rot 13

#include <unistd.h>

int main(int argc, char \*\*argv)

{

char c;

if (argc == 2)

{

while (\*argv[1])

{

c = \*argv[1];

if ((\*argv[1] >= 'a' && \*argv[1] < 'n') ||

(\*argv[1] >= 'A' && \*argv[1] < 'N'))

c += 13;

else if ((\*argv[1] >= 'n' && \*argv[1] <= 'z') ||

(\*argv[1] >= 'N' && \*argv[1] <= 'Z'))

c -= 13;

write (1, &c, 1);

argv[1]++;

}

}

write (1, "\n", 1);

return (0);

}

Ulstr

#include <unistd.h>

int main(int argc, char \*\*argv)

{

if (argc == 2)

{

while (\*argv[1])

{

if (\*argv[1] >= 'a' && \*argv[1] <= 'z')

\*argv[1] -= 32;

else if (\*argv[1] >= 'A' && \*argv[1] <= 'Z')

\*argv[1] += 32;

write (1, argv[1], 1);

argv[1]++;

}

}

write (1, "\n", 1);

}

Rot one

#include <unistd.h>

void ft\_putchar(char c)

{

write (1, &c, 1);

}

void rot\_one(char \*s)

{

while (\*s)

{

if ((\*s >= 'A' && \*s <= 'Y') || (\*s >= 'a' && \*s <= 'y'))

ft\_putchar(\*s + 1);

else if (\*s == 'Z' || \*s == 'z')

ft\_putchar(\*s - 25);

else

ft\_putchar(\*s);

++s;

}

}

int main(int argc, char \*\*argv)

{

if (argc == 2)

rot\_one(argv[1]);

ft\_putchar('\n');

return (0);

}

Swap

#include <unistd.h>

void ft\_swap(int \*a, int \*b)

{

int tmp;

tmp = \*a;

\*a = \*b;

\*b = tmp;

}

Repeat alpha

#include <unistd.h>

int letter\_count(char c)

{

int repeat;

if (c >= 'A' && c <= 'Z')

repeat = c - 'A' + 1;

else if (c >= 'a' && c <= 'z')

repeat = c - 'a' + 1;

else

repeat = 1;

return (repeat);

}

int main(int ac, char \*\*av)

{

int repeat;

if (ac == 2)

{

while (\*av[1])

{

repeat = letter\_count(\*av[1]);

while (repeat--)

write(1, av[1], 1);

av[1]++;

}

}

ft\_putchar('\n');

}