ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Λειτουργία Συστήματα Υπολογιστών

Αναφορά για την 1η Εργαστηριακή Άσκηση: Εισαγωγή στο περιβάλλον προγραμματισμού

Ον/μο :Μεσσής Κωνσταντίνος
Α.Μ.: :03111122

Ον/μο :Μπερέτσος Θεόδωρος
Α.Μ.: :03111612

Ημερομηνία παράδοσης: 20/11/2014

1.1 Σύνδεση με αρχείο αντικειμένων

-Πηγαίος κώδικας (source code)

main.c

```
#include "zing.h"

int main(void){
	zing();
	return 0;
}
```

zing2.c

```
#include <unistd.h>
#include <stdio.h>

int zing(){
	printf("Hello %s!!!\n", getlogin());
	return 0;
}
```

-Διαδικασία μεταγλώττισης και σύνδεσης

Όπως η άσκηση ζητά ορίζουμε νέο πηγαίο κώδικα main.c ο οποίος περιέχει τη συνάρτηση main() που καλεί τη συνάρτηση zing() όπως αυτή ορίζεται στο zing.o Στη συνέχεια κάνουμε compile το main.c με την εξής εντολή:

```
gcc –Wall –c main.c
```

από την οποία προκύπτει το μεταγλωττισμένο αρχείο main.o το οποίο συνέχεια συνδέουμε με το zing.o για να προκύψει το εκτελέσιμο με όνομα zing σύμφωνα με την εντολή:

```
gcc –o main.o zing.o zing
```

Ομοίως γράφουμε νέο πηγαίο κώδικα zing2.c οποίος ορίζει ξανά τη συνάρτηση zing(), τον μεταγλωττίζουμε με την εντολή:

```
gcc –Wall –c zing2.c
```

από την οποία προκύπτει το μεταγλωττισμένο αρχείο zing2.o το οποίο συνέχεια συνδέουμε με το main.o για να προκύψει το εκτελέσιμο με όνομα zing2 σύμφωνα με την εντολή:

```
gcc –o main.o zing2.o zing2
```

-Έξοδο εκτέλεσης του προγράμματος

Η εντολή ./zing έχει ως έξοδο στην οθόνη το εξής μήνυμα:

```
oslabb15@lesvos:~/oslab_ex1$ ./zing
Hello oslabb15!
```

Η εντολή ./zing2 έχει ως έξοδο στην οθόνη το εξής μήνυμα:

```
oslabb15@lesvos:~/oslab_ex1$ ./zing2
Hello oslabb15!!!
```

-Σύντομες απαντήσεις στις ερωτήσεις

Ερώτηση1: Ποιο σκοπό εξυπηρετεί η επικεφαλίδα;

Απάντηση:

Μία επικεφαλίδα είναι μια διεπαφή προς άλλα κομμάτια κώδικα που περιέχει πρότυπα και δηλώσεις. Στην περίπτωσή μας η εντολή

```
#include "zing.h"
```

που περιέχεται στους πηγαίους κώδικες που καλούν τη συνάρτηση zing() χρησιμεύει αντί της αντιγραφής του κώδικα της συνάρτησης στον αντίστοιχο κώδικα.

Ερώτηση 2: Ζητείται κατάλληλο Makefile για τη δημιουργία του εκτελέσιμου της άσκησης.

Απάντηση:

```
main.o: main.c
	gcc -Wall -c main.c
zing: zing.o main.o
	gcc -o zing zing.o main.o
```

Ερώτηση 3: Γράψτε το δικό σας zing2.o, το οποίο θα περιέχει zing() που θα εμφανίζει διαφορετικό αλλά παρόμοιο μήνυμα με τη zing() του zing.o. Συμβουλευτείτε το manual page της getlogin(3). Αλλάξτε το Makefile ώστε να παράγονται δύο εκτελέσιμα, ένα με το zing.o, ένα με το zing2.o, επαναχρησιμοποιώντας το κοινό object file main.o.

Απάντηση:

Ο πηγαίος κώδικας zing2.c παρατέθηκε παραπάνω.

```
main.o: main.c
	gcc -Wall -c main.c
zing: zing.o main.o
	gcc -o zing zing.o main.o
zing2.o : zing2.c
	gcc -Wall -c zing2.c
zing2: zing2.o main.o
	gcc -o zing2 zing2.o main.o
```

Ερώτηση 4: Έστω ότι έχετε γράψει το πρόγραμμά σας σε ένα αρχείο που περιέχει 500 συναρτήσεις. Αυτή τη στιγμή κάνετε αλλαγές μόνο σε μία συνάρτηση. Ο κύκλος εργασίας είναι: αλλαγές στον κώδικα, μεταγλώττιση, εκτέλεση, αλλαγές στον κώδικα, κ.ο.κ. Ο χρόνος μεταγλώττισης είναι μεγάλος, γεγονός που σας καθυστερεί. Πώς μπορεί να αντιμετωπισθεί το πρόβλημα αυτό;

Απάντηση:

Το πρόβλημα αυτό μπορεί να αντιμετωπισθεί με διαχωρισμό της κάθε συνάρτησης σε ξεχωριστό αρχείο πηγαίου κώδικα έτσι ώστε να μπορούμε να μεταβάλλουμε το περιεχόμενο κάποιας συνάρτησης και να την μεταγλωττίζουμε ξεχωριστά, χωρίς να χρειάζεται εκ νέου μεταγλώττιση και των υπολοίπων, χρησιμοποιώντας πάντα την ίδια εντολή για τη σύνδεση(linking) τους.

Ερώτηση 5: Ο συνεργάτης σας και εσείς δουλεύατε στο πρόγραμμα foo.c όλη την προηγούμενη εβδομάδα. Καθώς κάνατε ένα διάλειμμα και ο συνεργάτης σας δούλευε στον κώδικα, ακούτε μια απελπισμένη κραυγή. Ρωτάτε τι συνέβει και ο συνεργάτης σας λέει ότι το αρχείο foo.c χάθηκε! Κοιτάτε το history του φλοιού και η τελευταία εντολή ήταν η:

```
gcc -Wall -o foo.c foo.c
```

Τι συνέβη;

Απάντηση:

Προφανώς ο συνεργάτης μας στην προσπάθεια να μεταγλωττίσει τον πηγαίο κώδικα foo.c, εκτέλεσε κατά λάθος την παραπάνω εντολή η οποία δημιούργησε ένα εκτελέσιμο με όνομα foo.c το οποίο έγραψε πάνω στο παλιό αρχείο πηγαίου κώδικα.

1.2 Συνένωση δύο αρχείων σε τρίτο

-Πηγαίος κώδικας (source code)

fconc.c

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int fconc(int argc, char **argv){
	int fd3;
	if (argc == 3){
		fd3 = open("fconc.out", O_CREAT | O_WRONLY, S_IRUSR | S_IWUSR);
		write_file(fd3, argv[1]);
		write_file(fd3, argv[2]);
		close(fd3);
		return 0;
	}
	if (argc == 4 && strcmp(argv[1], argv[3]) != 0 && strcmp(argv[2], argv[3]
!= 0)){
		fd3 = open(argv[3], O_CREAT | O_WRONLY, S_IRUSR | S_IWUSR);
		write_file(fd3, argv[1]);
		write_file(fd3, argv[2]);
		close(fd3);
		return 0;
	}
	else{
		printf("Usage: ./fconc infile1 infile2 [outfile
(default:fconc.out)]");
		return 0;
	}
}
```

doWrite.c

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
```

```c
#include <stdlib.h>
#include <unistd.h>

void doWrite(int fd, const char *buff, int len){
	ssize_t wcnt;
	int remain = len;
	int idx = 0;
	do {
		wcnt = write(fd, buff + idx, remain);
		if (wcnt < 0){ /* error */
			perror("write");
			return;
		}
		idx += wcnt;
		remain -= idx;
	} while (idx < len);
	return;
}
```

WriteFile.c

```c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

void write_file(int fd, const char *infile){
	int fd2;
	char buff[1024];
	intt rcnt = 1;
	fd2 = open(infile, O_RDONLY);
	if (fd2 == -1){
		perror(infile);
		exit(1);
	}
	while (rcnt != 0) {
		rcnt = read(fd2, buff, sizeof(buff)-1);
		if (rcnt == -1){ /* error */
			perror("read");
			return;
		}
		if (rcnt != 0 ){
			buff[rcnt] = '\0';
			doWrite(fd, buff, rcnt + 1);
		}
	}
	close(fd2);
}
```

Αφού μεταγλωττίσουμε το καθένας από τα παραπάνω κομμάτια πηγαίου κώδικα με τις εντολές:

```
gcc -Wall -c fconc.c
gcc -Wall -c doWrite.c
gcc -Wall -c WriteFile.c
```

και έπειτα τα συνδέουμε (link) με την εντολή:

```
gcc fconc.o doWrite.o WriteFile.o –o fconc
```

-Σύντομες απαντήσεις στις ερωτήσεις

Ερώτηση 1: Εκτελέστε ένα παράδειγμα του fconc χρησιμοποιώντας την εντολή strace. Αντιγράψτε το κομμάτι της εξόδου της strace που προκύπτει από τον κώδικα που γράψατε.

Απάντηση:

Η `strace` παρουσιάζει τις κλήσεις συστήματος που γίνονται κατά την εκτέλεση της `fconc`. Αυτές που καλούνται από το δικό μας πρόγραμμα είναι οι `open, close, read` και `write,` όπως φαίνεται παραπάκω:

```
oslabb15@lesvos:~/oslab_ex1$ strace ./fconc A C
execve("./fconc", ["./fconc", "A", "C"], [/* 18 vars */]) = 0
brk(0)                                  = 0x804a000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb770e000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=67766, ...}) = 0
mmap2(NULL, 67766, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb76fd000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/i686/cmov/libc.so.6", O_RDONLY) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\260l\1\0004\0\0\0\34"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1331684, ...}) = 0
mmap2(NULL, 1337704, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb75b6000
mmap2(0xb76f7000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x141)
= 0xb76f7000
mmap2(0xb76fa000, 10600, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
xb76fa000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb75b5000
set_thread_area({entry_number:-1 -> 6, base_addr:0xb75b56c0, limit:1048575, seg_32bit:1, conte
nts:0, read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
mprotect(0xb76f7000, 8192, PROT_READ)   = 0
mprotect(0xb772c000, 4096, PROT_READ)   = 0
munmap(0xb76fd000, 67766)               = 0
open("fconc.out", O_WRONLY|O_CREAT, 0600) = 3
open("A", O_RDONLY)                     = 4
read(4, "PATRAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"..., 1023) = 35
write(3, "PATRAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"..., 36) = 36
read(4, ""..., 1023)                    = 0
close(4)                                = 0
open("C", O_RDONLY)                     = 4
read(4, "text1\n\0text2\n\0"..., 1023)  = 14
write(3, "text1\n\0text2\n\0\0"..., 15) = 15
read(4, ""..., 1023)                    = 0
close(4)                                = 0
close(3)                                = 0
exit_group(0)                           = ?
```