

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Εργασία 1

Ονοματεπώνυμο

AM

Χαραλαμπίδης Γεώργιος

1115201600193

Οδηγίες μεταγλώττισης του προγράμματος

Η μεταγλώττιση γίνεται με την εντολή `make`, αφού πρώτα βεβαιωθούμε ότι βρισκόμαστε στο `directory` του `project`.

Με την εντολή `make clean`, διαγράφονται τα αρχεία `object` και `executable` που δημιουργήσαμε με την `make`.

Τα `executables` είναι τα `process_a` και `process_b`.

Οδηγίες χρήσης του προγράμματος

Ανοίγουμε δύο διαφορετικά `terminal`.

Πληκτρολογούμε `make` αφού βεβαιωθούμε ότι βρισκόμαστε στο σωστό `directory`.

Έπειτα τρέχουμε:

`./process_a` σε `terminal 1`

`./process_b` σε `terminal 2`

Υπάρχουν επιλογές για τα `flags` (`-debug` και `-file`).

Παραδείγματα εκτέλεσης και περισσότερες πληροφορίες θα βρείτε στις σελίδες **4-6**.

1. Εισαγωγή και Σκοπός των Προγραμμάτων

Τα προγράμματα `process_a.c` και `process_b.c` υλοποιούν μια απλή επικοινωνία δύο διεργασιών (`processes`) μέσω κοινόχρηστης μνήμης (`shared memory`). Το κυρίως έργο τους είναι να ανταλλάσσουν μηνύματα, χωρίζοντάς τα σε τμήματα και επανασυνθέτοντάς τα στη συνέχεια.

2. Κύριες Λειτουργίες και Υλοποίηση

a. Δημιουργία και Διαχείριση Κοινόχρηστης Μνήμης

- Δημιουργία Κοινόχρηστης Μνήμης: Χρησιμοποιείται η συνάρτηση ``shm_open`` για τη δημιουργία ή σύνδεση με κοινόχρηστη μνήμη.
- Αντιστοίχιση Μνήμης: Η συνάρτηση ``mmap`` αντιστοιχίζει την κοινόχρηστη μνήμη στον χώρο διευθύνσεων της διεργασίας.

b. Διαχείριση Μηνυμάτων

- Τμηματοποίηση Μηνυμάτων: Κάθε μήνυμα χωρίζεται σε τμήματα, τα οποία αποθηκεύονται στην κοινόχρηστη μνήμη.
- Επανασύνθεση Μηνυμάτων: Τα τμήματα επανασυνθέτονται για να δημιουργηθεί το αρχικό μήνυμα.

c. Συγχρονισμός με Σηματοφόρους

- Σηματοφόροι (Semaphores): Χρησιμοποιούνται για να συγχρονίζουν τις διεργασίες στην πρόσβαση της κοινόχρηστης μνήμης.

3. Ανάλυση των Νημάτων και Συναρτήσεων

a. Διαχείριση Νημάτων (Threads)

Και τα δύο προγράμματα δημιουργούν δύο νήματα: ένα για την αποστολή και ένα για τη λήψη μηνυμάτων. Τα νήματα χρησιμοποιούνται για να επιτρέψουν ταυτόχρονες λειτουργίες αποστολής και λήψης, βελτιώνοντας έτσι την αποδοτικότητα της διαδικασίας επικοινωνίας.

- Νήμα Αποστολής (``send_message``): Αναλαμβάνει την ανάγνωση εισερχόμενων μηνυμάτων από τον χρήστη, την τμηματοποίησή τους, και την αποστολή τους μέσω κοινόχρηστης μνήμης.
- Νήμα Λήψης (``receive_message``): Παρακολουθεί την κοινόχρηστη μνήμη για τη λήψη τμηματοποιημένων μηνυμάτων, τα επανασυνθέτει και τα εμφανίζει στον χρήστη.

b. Χρήση Σηματοφόρων (Semaphores)

Για τον συγχρονισμό των νημάτων και την ασφαλή πρόσβαση στην κοινόχρηστη μνήμη, τα προγράμματα χρησιμοποιούν δύο σηματοφόρους (``sem_t sem1``, ``sem_t sem2``).

- ``sem1``: Χρησιμοποιείται για να εξασφαλίσει ότι ένα νήμα μπορεί να γράψει στην κοινόχρηστη μνήμη πριν το άλλο νήμα αρχίσει να διαβάζει από αυτήν.
- ``sem2``: Χρησιμοποιείται για να ειδοποιήσει το νήμα λήψης ότι υπάρχουν δεδομένα προς λήψη.

Αυτός ο μηχανισμός συγχρονισμού εξασφαλίζει ότι τα δεδομένα δεν θα διαβάζονται και γράφονται ταυτόχρονα, αποφεύγοντας πιθανά race conditions.

c. Τμηματοποίηση και Επανασύνθεση Μηνυμάτων

- Συνάρτηση `segment_message`: Διασπά το μήνυμα σε τμήματα του προκαθορισμένου μεγέθους `SEGMENT_SIZE`, αποθηκεύοντας τα στην κοινόχρηστη μνήμη.
- Συνάρτηση `reassemble_message`: Επανασυνθέτει το μήνυμα από τα τμήματα που έχουν αποθηκευτεί στην κοινόχρηστη μνήμη.

d. Αντιμετώπιση Τερματισμού και Εκκαθάρισης Πόρων

Και τα δύο προγράμματα χειρίζονται επίσης τον τερματισμό της επικοινωνίας και την εκκαθάριση των πόρων που χρησιμοποιήθηκαν.

- **Τερματισμός Επικοινωνίας**: Όταν λάβουν ένα συγκεκριμένο σήμα (π.χ., το μήνυμα `"#BYE#\n"`), οι διεργασίες θέτουν μία σημαία τερματισμού (`termination_flag`). Αυτό ειδοποιεί τα νήματα ότι πρέπει να σταματήσουν την επεξεργασία και να προχωρήσουν στην εκκαθάριση.
- **Εκκαθάριση Πόρων**: Περιλαμβάνει την καταστροφή των σηματοφόρων, την αποδέσμευση της κοινόχρηστης μνήμης μέσω `munmap`, και το κλείσιμο του αντίστοιχου αρχείου κοινόχρηστης μνήμης με `shm_unlink`.

e. Λειτουργία Σε Debug Mode

Υπάρχει επίσης η δυνατότητα εκκίνησης των προγραμμάτων σε "debug mode", όπου εμφανίζονται επιπλέον εκτυπώσεις για διαγνωστικούς σκοπούς. Αυτό γίνεται μέσω της παραμέτρου `-debug` στη γραμμή εντολών.

f. Δυνατότητα αποθήκευσης των αποτελεσμάτων σε αρχείο

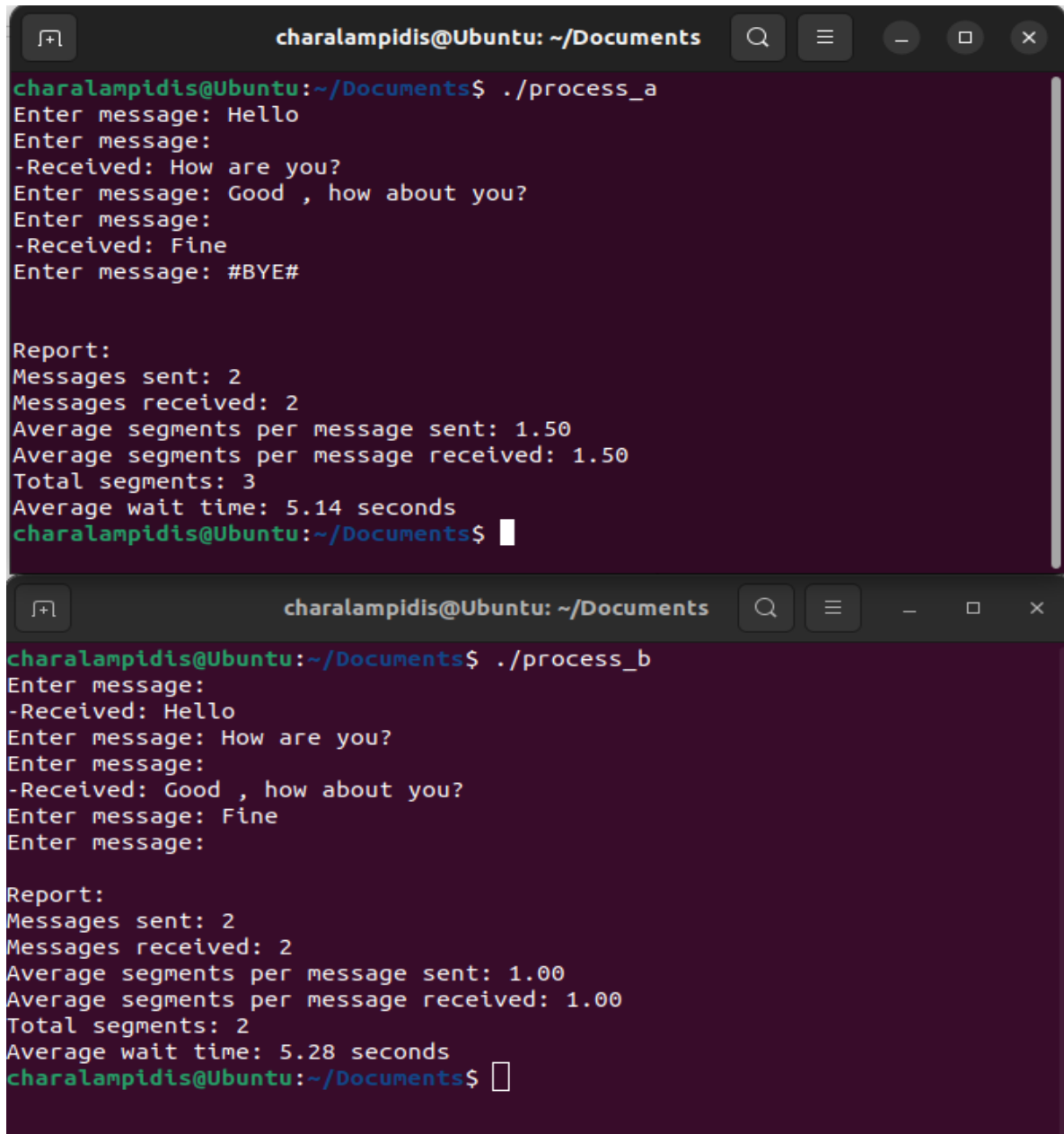
Πέρα από την εντολή `-debug` κατά την εκτέλεση του προγράμματος, υπάρχει και η `-file <enter path to file>` όπου με την χρήση της αποθηκεύονται όλα τα `printfs` των προγραμμάτων στα αντίστοιχα αρχεία που έχουμε ορίσει.

Παρακάτω παραθέτω την αλληλεπίδραση των προγραμμάτων:

Βασική εκτέλεση των προγραμμάτων:

`./process_a`

`./process_b`



The image shows two terminal windows from a Ubuntu system, both titled 'charalampidis@Ubuntu: ~/Documents'. The top window shows the execution of `./process_a`. It displays a sequence of messages sent and received, followed by a report. The bottom window shows the execution of `./process_b`, which also displays a sequence of messages and a report.

```
charalampidis@Ubuntu:~/Documents$ ./process_a
Enter message: Hello
Enter message:
-Received: How are you?
Enter message: Good , how about you?
Enter message:
-Received: Fine
Enter message: #BYE#

Report:
Messages sent: 2
Messages received: 2
Average segments per message sent: 1.50
Average segments per message received: 1.50
Total segments: 3
Average wait time: 5.14 seconds
charalampidis@Ubuntu:~/Documents$
```

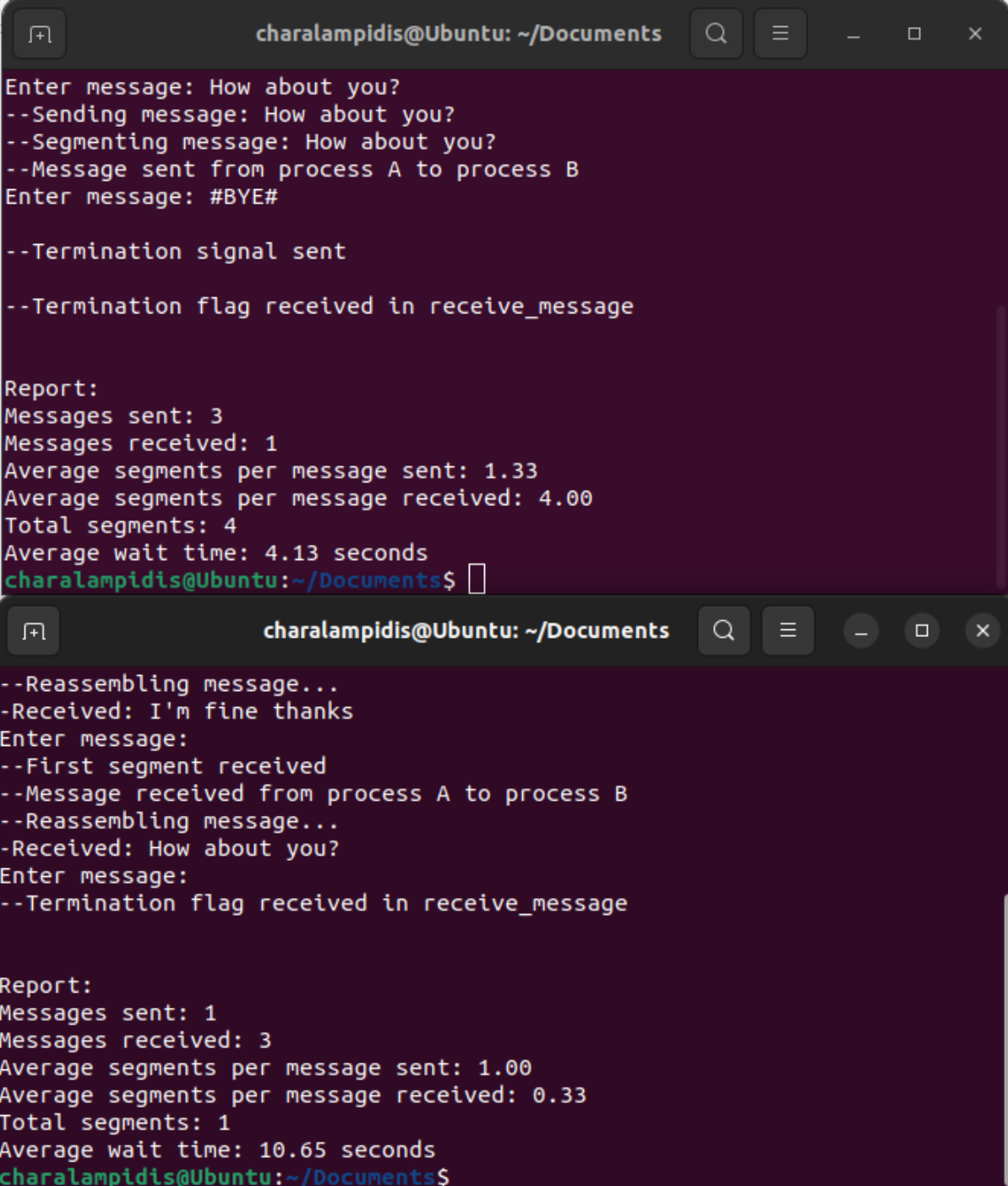
```
charalampidis@Ubuntu:~/Documents$ ./process_b
Enter message:
-Received: Hello
Enter message: How are you?
Enter message:
-Received: Good , how about you?
Enter message: Fine
Enter message:

Report:
Messages sent: 2
Messages received: 2
Average segments per message sent: 1.00
Average segments per message received: 1.00
Total segments: 2
Average wait time: 5.28 seconds
charalampidis@Ubuntu:~/Documents$
```

Εκτέλεση των προγραμμάτων με την χρήση flag -debug

```
./process_a -debug
```

```
./process_b -debug
```



The image shows two terminal windows side-by-side, both titled 'charalampidis@Ubuntu: ~/Documents'. The top window shows the output of 'process_a -debug'. It starts with a prompt 'Enter message: How about you?' and then displays several debug messages: '--Sending message: How about you?', '--Segmenting message: How about you?', '--Message sent from process A to process B', and 'Enter message: #BYE#'. It then shows '--Termination signal sent' and '--Termination flag received in receive_message'. Finally, it prints a 'Report:' with statistics: 'Messages sent: 3', 'Messages received: 1', 'Average segments per message sent: 1.33', 'Average segments per message received: 4.00', 'Total segments: 4', and 'Average wait time: 4.13 seconds'. The bottom window shows the output of 'process_b -debug'. It starts with '--Reassembling message...' and '-Received: I'm fine thanks'. Then it prompts 'Enter message:' and shows '--First segment received', '--Message received from process A to process B', and '--Reassembling message...'. It then shows '-Received: How about you?' and prompts 'Enter message:'. It then shows '--Termination flag received in receive_message'. Finally, it prints a 'Report:' with statistics: 'Messages sent: 1', 'Messages received: 3', 'Average segments per message sent: 1.00', 'Average segments per message received: 0.33', 'Total segments: 1', and 'Average wait time: 10.65 seconds'.

```
charalampidis@Ubuntu: ~/Documents
Enter message: How about you?
--Sending message: How about you?
--Segmenting message: How about you?
--Message sent from process A to process B
Enter message: #BYE#

--Termination signal sent

--Termination flag received in receive_message

Report:
Messages sent: 3
Messages received: 1
Average segments per message sent: 1.33
Average segments per message received: 4.00
Total segments: 4
Average wait time: 4.13 seconds
charalampidis@Ubuntu:~/Documents$

charalampidis@Ubuntu: ~/Documents
--Reassembling message...
-Received: I'm fine thanks
Enter message:
--First segment received
--Message received from process A to process B
--Reassembling message...
-Received: How about you?
Enter message:
--Termination flag received in receive_message

Report:
Messages sent: 1
Messages received: 3
Average segments per message sent: 1.00
Average segments per message received: 0.33
Total segments: 1
Average wait time: 10.65 seconds
charalampidis@Ubuntu:~/Documents$
```

Εκτέλεση των προγραμμάτων με την χρήση flag -file

`./process_a -file resultsA.txt`

`./process_b -file resultsB.txt`

The image shows two screenshots of a terminal window on an Ubuntu system, demonstrating the execution of two programs, `process_a` and `process_b`, with the `-file` flag to save their output to text files.

Top Screenshot: Execution of `process_a`

The terminal shows the user running `./process_a -file resultsA.txt`. The program prompts for a message, receives it, and then prints a report. The output is saved to `resultsA.txt`.

```
charalampidis@Ubuntu: ~/Documents
charalampidis@Ubuntu:~/Documents$ ./process_a -file resultsA.txt
Hi
How are you?
I'm fine as well
charalampidis@Ubuntu:~/Documents$
```

The output file `resultsA.txt` contains the following text:

```
1 Enter message: Enter message:
2 -Received: Hello
3 Enter message: Enter message:
4 -Received: I'm good , you?
5 Enter message: Enter message:
6 -Received: That's good to hear
7 Enter message:
8
9 Report:
10 Messages sent: 3
11 Messages received: 3
12 Average segments per message sent: 1.33
13 Average segments per message received: 1.33
14 Total segments: 4
15 Average wait time: 6.20 seconds
```

Bottom Screenshot: Execution of `process_b`

The terminal shows the user running `./process_b -file resultsB.txt`. The program prompts for a message, receives it, and then prints a report. The output is saved to `resultsB.txt`.

```
charalampidis@Ubuntu: ~/Documents
charalampidis@Ubuntu:~/Documents$ ./process_b -file resultsB.txt
Hello
I'm good , you?
That's good to hear
#BYE#
charalampidis@Ubuntu:~/Documents$
```

The output file `resultsB.txt` contains the following text:

```
1 Enter message:
2 -Received: Hi
3 Enter message: Enter message:
4 -Received: How are you?
5 Enter message: Enter message:
6 -Received: I'm fine as well
7 Enter message: Enter message:
8
9 Report:
10 Messages sent: 3
11 Messages received: 3
12 Average segments per message sent: 1.67
13 Average segments per message received: 1.67
14 Total segments: 5
15 Average wait time: 5.28 seconds
```