ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ (2007-08)

Άσκηση 4

Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται `packing.c`) το οποίο να είναι σε θέση να "πακετάρει" σ' ένα καθολικό αρχείο τα περιεχόμενα δεδομένων αρχείων, αλλά και να μπορεί να εξάγει από το καθολικό αρχείο τα επί μέρους αρχεία που έχουν φυλαχθεί σ' αυτό. Για να είναι δυνατή η ανάκτηση των αρχείων από το καθολικό αρχείο, το τελευταίο έχει συγκεκριμένη δομή που περιγράφεται στη συνέχεια.

Τα πρώτα 2048 bytes του καθολικού αρχείου αποτελούν το ευρετήριό του, υπό την έννοια ότι εκεί περιέχονται πληροφορίες για τα αρχεία που έχουν καταχωρηθεί σ' αυτό. Το μέγεθος του ευρετηρίου (2048) να ορισθεί στο πρόγραμμά σας με συμβολική σταθερά (μέσω `#define`). Θεωρήστε ότι το ευρετήριο αποτελείται από ομάδες των 32 bytes, ας τις λέμε στο εξής *εγγραφές*, όπου κάθε εγγραφή αντιστοιχεί σ' ένα αρχείο που είναι καταχωρημένο στο καθολικό αρχείο. Αυτό σημαίνει ότι το πολύ 64 (=2048/32) αρχεία είναι δυνατόν να πακεταρισθούν μέσα στο καθολικό αρχείο. Σε μία εγγραφή στο ευρετήριο καταχωρείται το μέγεθος του αρχείου (σε bytes) και το όνομά του. Μπορούμε να περιγράψουμε τις πληροφορίες που περιέχονται σε μία εγγραφή του ευρετηρίου με τη δομή:[1]

```
struct headrec {
   int size;
   char filename[28];
};
```

Τα αρχεία είναι καταχωρημένα στο καθολικό αρχείο κατά σειρά αμέσως μετά το ευρετήριο. Κάθε αρχείο αρχίζει στο αμέσως επόμενο byte από αυτό που τελειώνει το προηγούμενό του (το πρώτο αρχείο αρχίζει αμέσως μετά το ευρετήριο) και εκτείνεται για `size` στο πλήθος bytes. Το `filename` είναι το όνομα του καταχωρημένου αρχείου για την αντίστοιχη εγγραφή. Όπως παρατηρείτε, δεν μπορεί να ξεπερνά τους 27 χαρακτήρες (αφού χρειάζεται και το `'\0'` για να τερματίσει η συμβολοσειρά όπως πρέπει). Οι χαρακτήρες που δεν χρησιμοποιούνται μέσα στο `filename` πρέπει να έχουν τεθεί σε `'\0'`. Αν το καθολικό αρχείο περιέχει λιγότερα από 64 αρχεία, τότε όλα τα bytes της επόμενης εγγραφής από την τελευταία στην οποία αντιστοιχεί ένα αρχείο μέσα στο καθολικό αρχείο, πρέπει να έχουν τεθεί σε 0.

Το πρόγραμμα που θα γράψετε θα πρέπει να μπορεί να κληθεί με έναν από τους παρακάτω τρόπους:

`packing` [`-v <packetfilename>`] `-p <filename>1 <filename>2 ... <filename>N`

`packing` [`-v <packetfilename>`] `-l`

`packing` [`-v <packetfilename>`] `-u <filename>1 <filename>2 ... <filename>K`

`packing` [`-v <packetfilename>`] `-u`

Το `<packetfilename>` είναι το όνομα του καθολικού αρχείου, αλλά το γεγονός ότι έχουμε περικλείσει την επιλογή `-v <packetfilename>` μέσα σε [...] σημαίνει ότι είναι προαιρετική. Αν δεν δίνεται, να θεωρείται σαν όνομα καθολικού αρχείου το `./packet.pck`.

Στη συνέχεια, εξηγούνται αναλυτικά οι δυνατοί τρόποι εκτέλεσης τους προγράμματος, που αναφέρθηκαν προηγουμένως.

[1] Θεωρήστε ότι στο σύστημα που δουλεύετε οι ακέραιοι (`int`) φυλάσσονται σε 4 bytes.

Με την κλήση

`packing [-v <packetfilename>] -p <filename>1 <filename>2 ... <filename>N`

να δημιουργείται το καθολικό αρχείο `<packetfilename>`, ή το `./packet.pck`, ανάλογα αν έχει δοθεί η επιλογή `-v` ή όχι, και να πακετάρονται μέσα σ' αυτό τα αρχεία `<filename>1`, `<filename>2`, ..., `<filename>N`. Αν το καθολικό αρχείο υπάρχει ήδη, ξαναγράφεται εξ αρχής, δηλαδή διαγράφονται τα προηγούμενα περιεχόμενά του κατά την έναρξη της διαδικασίας. Καθένα από τα `<filename>i` μπορεί να είναι ένα όνομα-μονοπάτι (απόλυτο ή σχετικό). Στην περίπτωση αυτή, μέσα στη σχετική εγγραφή στο ευρετήριο του καθολικού αρχείου θα καταχωρηθεί μόνο το όνομα του αρχείου, αφού αποκοπεί το τυχόν μονοπάτι που προηγείται. Αν το όνομα του αρχείου ξεπερνά τους 27 χαρακτήρες (μετά την αποκοπή του πιθανού μονοπατιού) να καταχωρούνται στο ευρετήριο μόνο οι πρώτοι 27 χαρακτήρες του ονόματος. Σε περίπτωση οποιουδήποτε λάθους (π.χ. μη υπαρκτό αρχείο, αδυναμία ανάγνωσης του αρχείου, κλπ.) το πακετάρισμα του συγκεκριμένου αρχείου να μην γίνεται, εκτυπώνοντας και κατάλληλο μήνυμα, αλλά να συνεχίζει κανονικά η διαδικασία για τα άλλα αρχεία.

Με την κλήση

`packing [-v <packetfilename>] -l`

να εκτυπώνονται στην έξοδο τα ονόματα των αρχείων που είναι καταχωρημένα μέσα στο καθολικό αρχείο, αλλά και το μέγεθος (σε bytes) του καθενός.

Με την κλήση

`packing [-v <packetfilename>] -u <filename>1 <filename>2 ... <filename>K`

να εξάγονται μέσα από το καθολικό αρχείο τα αρχεία με ονόματα `<filename>1`, `<filename>2`, ..., `<filename>K`, ενώ με την κλήση

`packing [-v <packetfilename>] -u`

να εξάγονται όλα τα αρχεία που περιέχονται μέσα στο καθολικό αρχείο. Η διαδικασία εξαγωγής αρχείων δεν επηρεάζει το καθολικό αρχείο. Σε περίπτωση που δεν υπάρχει αρχείο καταχωρημένο με όνομα κάποιο από τα `<filename>i`, να εκτυπώνεται κατάλληλο μήνυμα λάθους, αλλά η διαδικασία εξαγωγής να συνεχίζει κανονικά για τα άλλα αρχεία.

Δείτε στη συνέχεια κάποια παραδείγματα εκτελέσεων του προγράμματος.[2]

```
% ls -l text_file1.txt zipped_binary_file.zip dir1/text_file2.txt
-rw-------   1 ip       other         44 Dec 16 08:10 dir1/text_file2.txt
-rw-------   1 ip       other         21 Dec 16 08:06 text_file1.txt
-rw-------   1 ip       other        364 Dec 16 08:12 zipped_binary_file.zip
% cat text_file1.txt
This is a test file.
% cat dir1/text_file2.txt
And another file
to test program packing.c.
%
```

[2] Με την εντολή "`od`" του Unix μπορούμε να δούμε το "εσωτερικό" ενός αρχείου (κειμένου ή, γενικότερα, δυαδικού). Δώστε "`man od`" για περισσότερες πληροφορίες.

```
% ./packing -v testpack.pck -p text_file1.txt dir1/text_file2.txt
Packing text_file1.txt
Packing dir1/text_file2.txt
Done!
% ls -l testpack.pck
-rw-------   1 ip       other       2113 Dec 16 08:24 testpack.pck
% od -tu1c testpack.pck
0000000 000 000 000 021 116 101 120 116 095 102 105 108 101 049 046 116
         \0  \0  \0 025   t   e   x   t   _   f   i   l   e   1   .   t
0000020 120 116 000 000 000 000 000 000 000 000 000 000 000 000 000 000
          x   t  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000040 000 000 000 044 116 101 120 116 095 102 105 108 101 050 046 116
         \0  \0  \0   ,   t   e   x   t   _   f   i   l   e   2   .   t
0000060 120 116 000 000 000 000 000 000 000 000 000 000 000 000 000 000
          x   t  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000100 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
         \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
*
0004000 084 104 105 115 032 105 115 032 097 032 116 101 115 116 032 102
          T   h   i   s       i   s       a       t   e   s   t       f
0004020 105 108 101 046 010 065 110 100 032 097 110 111 116 104 101 114
          i   l   e   .  \n   A   n   d       a   n   o   t   h   e   r
0004040 032 102 105 108 101 010 116 111 032 116 101 115 116 032 112 114
              f   i   l   e  \n   t   o       t   e   s   t       p   r
0004060 111 103 114 097 109 032 112 097 099 107 105 110 103 046 099 046
          o   g   r   a   m       p   a   c   k   i   n   g   .   c   .
0004100 010
         \n
0004101
% ./packing -v testpack.pck -l
text_file1.txt (21 bytes)
text_file2.txt (44 bytes)
% cd dir2
% ls -l
total 0
% ./packing -v ../testpack.pck -u text_file2.txt
Unpacking text_file2.txt
Done!
% ls -l
-rw-------   1 ip       other         44 Dec 16 08:32 text_file2.txt
% diff text_file2.txt ../dir1/text_file2.txt
% ./packing -p ../text_file1.txt ../zipped_binary_file.zip
Packing ../text_file1.txt
Packing ../zipped_binary_file.zip
Done!
% ls -l packet.pck
-rw-------   1 ip       other       2433 Dec 16 08:35 packet.pck
%
```

```
% od -tu1c packet.pck
0000000 000 000 000 021 116 101 120 116 095 102 105 108 101 049 046 116
          \0  \0  \0 025   t   e   x   t   _   f   i   l   e   1   .   t
0000020 120 116 000 000 000 000 000 000 000 000 000 000 000 000 000 000
           x   t  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000040 000 000 001 108 122 105 112 112 101 100 095 098 105 110 097 114
          \0  \0 001   l   z   i   p   p   e   d   _   b   i   n   a   r
0000060 121 095 102 105 108 101 046 122 105 112 000 000 000 000 000 000
           y   _   f   i   l   e   .   z   i   p  \0  \0  \0  \0  \0  \0
0000100 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
          \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
*
0004000 084 104 105 115 032 105 115 032 097 032 116 101 115 116 032 102
           T   h   i   s       i   s       a       t   e   s   t       f
0004020 105 108 101 046 010 080 075 003 004 020 000 000 000 008 000 246
           i   l   e   .  \n   P   K 003 004 024  \0  \0  \0  \b  \0   f
0004040 076 099 052 222 062 117 146 186 000 000 000 185 001 000 000 023
           L   c   4   ’h   >   u 222   ’I  \0  \0  \0   ’H 001  \0  \0 027
0004060 000 021 000 104 111 109 101 047 117 115 101 114 115 047 116 097
          \0 025  \0   h   o   m   e   /   u   s   e   r   s   /   t   a
0004100 107 105 115 047 046 108 111 103 105 110 085 084 009 000 003 192
           k   i   s   /   .   l   o   g   i   n   U   T  \t  \0 003
.....................................................................
0004560 000 000 000 001 000 001 000 082 000 000 000 004 001 000 000 000
          \0  \0  \0 001  \0 001  \0   R  \0  \0  \0 004 001  \0  \0  \0
0004600 000
          \0
0004601
% ./packing -l
text_file1.txt (21 bytes)
zipped_binary_file.zip (364 bytes)
% ./packing -u
Unpacking text_file1.txt
Unpacking zipped_binary_file.zip
Done!
% ls -l text_file1.txt zipped_binary_file.zip
-rw-------   1 ip       other         21 Dec 16 08:38 text_file1.txt
-rw-------   1 ip       other        364 Dec 16 08:38 zipped_binary_file.zip
% cmp text_file1.txt ../text_file1.txt
% cmp zipped_binary_file.zip ../zipped_binary_file.zip
%
```

Η παράδοση της άσκησης αυτής συνίσταται στην υποβολή του πηγαίου αρχείου `packing.c` με διαδικασία που θα ανακοινωθεί σύντομα.