Θέτοντας και επιστρέφοντας την τιμή της προτεραιότητας διεργασίας

Το επίπεδο προτεραιότητας μιας διεργασίας μπορεί να αλλάξει με χρήση της συνάρτησης nice. Κάθε διεργασία διαθέτει μια τιμή που καλείται nice που χρησιμοποιείται για να υπολογίζει το επίπεδο προτεραιότητας της καλούμενης διεργασίας. Μία διεργασία κληρονομεί την προτεραιότητα της διεργασίας που τη δημιούργησε. Η προτεραιότητα μιας διεργασίας μπορεί να υποβαθμιστεί αυξάνοντας την τιμή nice. Μόνο οι διεργασίες διαχειριστή και πυρήνα μπορούν να αναβαθμίσουν την προτεραιότητά τους. Η δήλωση της συνάρτησης nice είναι η εξής:

#include <unistd.h>

int nice(int incr);

Μια χαμηλή τιμή της nice αυξάνει το επίπεδο προτεραιότητας της διεργασίας. Η παράμετρος incr είναι η τιμή που προστίθεται στην τρέχουσα τιμή της nice για την καλούμενη διεργασία. Η παράμετρος incr μπορεί να είναι αρνητική ή θετική. Η τιμή της nice όμως είναι μη αρνητικός αριθμός. Η θετική τιμή του incr αυξάνει την τιμή της nice, άρα μειώνει το επίπεδο προτεραιότητας. Μια αρνητική τιμή του incr θα μειώσει την τιμή της nice, αυξάνοντας έτσι την προτεραιότητα. Εάν η τιμή της incr τροποποιήσει την τιμή της nice πάνω ή κάτω από τα όρια, τότε η τιμή της nice τίθεται στο υψηλότερο ή χαμηλότερο όριο, αντίστοιχα. Όταν επιστρέφει επιτυχώς η συνάρτηση nice επιστρέφει τη νέα τιμή nice της διεργασίας. Αν δεν επιστρέψει με επιτυχία, η συνάρτηση θα επιστρέψει –1 και η τιμή της nice δεν αλλάζει.

#include <sys/resource.h>

int getpriority(int which, id_t who);

int setpriority(int which, id_t who, int value);

Η συνάρτηση setpriority θέτει την τιμή nice μιας διεργασίας, μιας ομάδας διεργασιών ή ενός χρήστη. Η συνάρτηση getpriority επιστρέφει την προτεραιότητα μιας διεργασίας, μιας ομάδας διεργασιών ή ενός χρήστη. Η παράμετρος which καθορίζει μια διεργασία, μια ομάδα διεργασιών ή ένα χρήστη. Μπορεί να πάρει τις ακόλουθες τιμές:

PRIO_PROCESS, καθορίζει μια διεργασία

PRIO_PGRP, καθορίζει μια ομάδα διεργασιών

PRIO_USER, καθορίζει ένα χρήστη

Με βάση την τιμή της παραμέτρου which, η τιμή της παραμέτρου who είναι ο αριθμός που αντιστοιχεί στην ταυτότητα (id number) μιας διεργασίας, μιας ομάδας διεργασιών ή ενός χρήστη. Η τιμή 0 της who υποδεικνύει την τρέχουσα διεργασία, την τρέχουσα ομάδα διεργασιών ή τον τρέχοντα χρήστη. Η τιμή της παραμέτρου value για τη συνάρτηση setpriority θα είναι η νέα τιμή της nice για τη συγκεκριμένη διεργασία, ομάδα διεργασιών ή χρήστη. Η τιμή της nice σε ένα περιβάλλον UNIX κινείται από –20 έως 19. Σε αντίθεση με τη συνάρτηση nice η τιμή που περνιέται στη setpriority είναι η πραγματική τιμή της nice που θέλουμε και όχι κάποιο offset που πρόκειται να προστεθεί στην τρέχουσα τιμή της nice.

Σε μια διεργασία με πολλαπλά νήματα, η τροποποίηση της προτεραιότητας θα επηρεάσει την προτεραιότητα όλων των νημάτων αυτής της διεργασίας. Σε περίπτωση επιτυχίας η getpriority θα επιστρέψει την τιμή nice για τη συγκεκριμένη διεργασία. Σε περίπτωση επιτυχίας η setpriority θα επιστρέψει την τιμή 0. Σε περίπτωση αποτυχίας και οι δύο συναρτήσεις επιστρέφουν την τιμή –1. Η τιμή –1 είναι μια νόμιμη τιμή nice για μία διεργασία. Για να αποφασιστεί εάν έχει συμβεί κάποιο σφάλμα πρέπει να ελεγχθεί η εξωτερική μεταβλητή errno.

Χρησιμοποιώντας την κλήση συστήματος fork()

Η κλήση συστήματος fork() δημιουργεί μια νέα διεργασία ως αντίγραφο της διεργασίας που εκτελεί την κλήση και καλείται γονική. Η fork επιστρέφει δύο τιμές εφόσον εκτελεστεί επιτυχώς, μία στη γονική διεργασία και μία στη θυγατρική διεργασία. Επιστρέφει 0 στην θυγατρική διεργασία και την ταυτότητα της νέας διεργασίας (PID) στην γονική διεργασία. Οι δύο διεργασίες συνεχίζουν να εκτελούνται από την επόμενη από την fork εντολή του προγράμματος. Εάν η fork αποτύχει, που σημαίνει ότι καμία θυγατρική διεργασία δεν δημιουργήθηκε, επιστρέφεται -1 στην γονική διεργασία.

#include <unistd.h>

pid_t fork(void);

Η fork () θα αποτύχει εφόσον το σύστημα δεν διαθέτει τους απαραίτητους πόρους για τη δημιουργία μιας άλλης διεργασίας. Εάν υπάρχει όριο στον αριθμό των διεργασιών που μια διεργασία μπορεί να δημιουργήσει ή υπάρχει όριο στον αριθμό των διεργασιών σε όλη την έκταση του συστήματος και το όριο αυτό έχει παραβιαστεί, η fork () θα αποτύχει. Στην περίπτωση αυτή το errno παίρνει την τιμή που δείχνει το ακριβές σφάλμα.

Χρησιμοποιώντας την οικογένεια των κλήσεων συστήματος exec()

Η οικογένεια των συναρτήσεων exec() αντικαθιστά την εικόνα της διεργασίας που κάνει την κλήση, με μία νέα εικόνα διεργασίας. Η fork () δημιουργεί μια νέα διεργασία ως πιστό αντίγραφο της γονικής διεργασίας, ενώ η συνάρτηση exec() αντικαθιστά την εικόνα της διεργασίας –αντίγραφο με μία νέα. Η εικόνα της νέας διεργασίας είναι ένα κανονικό εκτελέσιμο αρχείο εκτελείται αμέσως. Το εκτελέσιμο μπορεί να καθοριστεί είτε σαν μονοπάτι είτε σαν όνομα αρχείου. Οι συναρτήσεις αυτές μπορούν να περάσουν ορίσματα γραμμής εντολής στη νέα διεργασία. Περιβαλλοντικές μεταβλητές μπορούν επίσης να καθοριστούν. Δεν υπάρχει επιστρεφόμενη τιμή εφόσον η συνάρτηση αποτύχει επειδή η εικόνα της διεργασίας που έκανε την κλήση γράφεται από πάνω. Σε περίπτωση αποτυχίας επιστρέφεται -1 στη διεργασία που έκανε την κλήση exec().

Όλες οι exec () συναρτήσεις μπορούν να αποτύχουν κάτω από τις ακόλουθες συνθήκες:

- Απαγορεύεται η πρόσβαση στο εκτελέσιμο αρχείο
- Το αρχείο δεν υπάρχει
- Το αρχείο δεν είναι εκτελέσιμο

- Υπάρχουν προβλήματα με συμβολικούς συνδέσμους, δηλαδή όταν υπάρχει επανάληψη κατά τη χρήση συμβολικών συνδέσμων κατά την ανάλυση του μονοπατιού προς το εκτελέσιμο αρχείο, ή οι συμβολικοί σύνδεσμοι κάνουν το μονοπάτι προς το εκτελέσιμο αρχείο να είναι πολύ μεγάλο.

Οι συναρτήσεις exec() χρησιμοποιούνται μαζί με την fork(). Η fork() δημιουργεί και αρχικοποιεί μία θυγατρική διεργασία αντιγράφοντας την γονική. Η θυγατρική διεργασία τότε αντικαθιστά την εικόνα της εκτελώντας μια exec(). Το παρακάτω παράδειγμα δείχνει τη χρήση του συνδυασμού των fork και exec.

```
//…
RtValue = fork();
if(RtValue == 0) {
        execl("/path/direct", "direct", ".");
}
```

Στο παραπάνω παράδειγμα η συνάρτηση fork() καλείται και επιστρέφει μια τιμή που αποθηκεύεται στην μεταβλητή RtValue. Εάν η RtValue έχει την τιμή 0, τότε μιλάμε για τη θυγατρική διεργασία. Η execl() συνάρτηση καλείται τότε. Η πρώτη παράμετρος είναι το μονοπάτι προς το εκτελέσιμο αρχείο, η δεύτερη παράμετρος είναι η εντολή εκτέλεσης, και η τρίτη παράμετρος είναι το όρισμα. Το direct είναι ένα πρόγραμμα που εμφανίζει σε λίστα όλους τους καταλόγους και τους υποκαταλόγους ενός δοθέντος καταλόγου. Υπάρχουν έξι (6) παραλλαγές συναρτήσεων exec(), όπου καθεμία έχει τον δικό της τρόπο κλήσης και χρήση.

1. Συναρτήσεις execl()

Οι συναρτήσεις execl(), execle() execlp() περνούν το όρισμα της γραμμής εντολής σαν λίστα. Ο αριθμός των ορισμάτων γραμμής εντολής πρέπει να είναι γνωστός κατά την ώρα της μετάφρασης ώστε να είναι χρήσιμες αυτές οι συναρτήσεις.

- int execl (const char *path, const char *arg0, **…**/*, (char *) 0 */);

path είναι το μονοπάτι προς το εκτελέσιμο πρόγραμμα. Μπορεί να καθοριστεί είτε σαν απόλυτο μονοπάτι είτε σαν σχετικό μονοπάτι με βάση τον τρέχοντα κατάλογο. Τα επόμενα ορίσματα είναι η λίστα των ορισμάτων του εκτελέσιμου προγράμματος από το arg0 έως το argn. Μπορούν να υπάρχουν μέχρι n ορίσματα. Το τέλος της λίστας ακολουθείται από ένα δείκτη NULL.

- int execle (const char *path, const char *arg0, **…**/*, (char *) 0 *, char *const envp[] */);

Η συνάρτηση αυτή είναι ακριβώς ίδια με την execl με τη μόνη διαφορά ότι παίρνει μία επιπλέον παράμετρο, την envp[]. Η παράμετρος περιέχει το νέο περιβάλλον για τη νέα διεργασία. Η envp[] είναι ένας δείκτης σε ένα πίνακα που τερματίζεται με null και περιέχει strings που τερματίζονται με null. Κάθε string είναι της μορφής: name = value, όπου name είναι το όνομα της μεταβλητής περιβάλλοντος και value είναι το string που πρέπει να αποθηκευτεί. Η envp[] μπορεί να καθοριστεί με τον ακόλουθο τρόπο: char *const envp[]={"PATH=/opt/kde2:/sbin", "HOME=/home", NULL};

Οι PATH και HOME είναι οι μεταβλητές περιβάλλοντος στην περίπτωση αυτή.

- int execlp (const char *file, const char *arg0, **…**/*, (char *) 0 */);

file είναι το όνομα του εκτελέσιμου προγράμματος. Χρησιμοποιεί την PATH μεταβλητή περιβάλλοντος για να εντοπίσει τα εκτελέσιμα. Τα υπόλοιπα ορίσματα είναι όπως στην execl() συνάρτηση.

Ακολουθούν μερικά παραδείγματα σύνταξης των execl() συναρτήσεων χρησιμοποιώντας αυτά τα ορίσματα:

char *const args[] = {“direct”, “.”, NULL};

char *const envp[] = {“files=50”, NULL};

execl (“/path/direct”, “direct”, “.”, NULL) ;

execle (“/path/direct”, “direct”, “.”, NULL, envp) ;

execlp (“direct”, “direct”, “.”, NULL) ;

2. *Συναρτήσεις execv()*

Οι συναρτήσεις execv(), execve() και execvp() περνούν τα ορίσματα γραμμής εντολής σε ένα διάνυσμα δεικτών προς strings που τερματίζουν με NULL. Ο αριθμός των ορισμάτων γραμμής εντολής πρέπει να είναι γνωστός κατά την ώρα της μετάφρασης ώστε να είναι χρήσιμες οι συναρτήσεις αυτές. Το argv[0] είναι συνήθως η εντολή εκτέλεσης.

- int execv (const char *path, char *const arg[]);

path είναι το μονοπάτι προς το εκτελέσιμο αρχείο. Μπορεί να καθοριστεί είτε ως απόλυτο είτε ως σχετικό μονοπάτι ως προς τον τρέχοντα κατάλογο. Το επόμενο όρισμα είναι ένα διάνυσμα που τερματίζεται σε NULL και περιέχει τα ορίσματα γραμμής εντολής σαν strings που τερματίζονται σε NULL. Το arg[] μπορεί να καθοριστεί με τον ακόλουθο τρόπο: char *const arg[]={“traverse”, “.”, “>”, “1000”, NULL}; Παρακάτω φαίνεται ένα παράδειγμα κλήσης της execv():

execv (“traverse”, arg);

Στην περίπτωση αυτή το πρόγραμμα traverse εμφανίζει σε λίστα όλα τα αρχεία του τρέχοντος καταλόγου που είναι μεγαλύτερα από 1000 bytes.

- int execve (const char *path, char *const arg[], char *const envp[]);

Η συνάρτηση αυτή είναι ίδια ακριβώς με την execv, εκτός του ότι περιέχει μια επιπλέον παράμετρο, την envp[], που έχει περιγραφεί προηγουμένως.

- int execvp (const char *file, char *const arg[]);

file είναι το όνομα του εκτελέσιμου προγράμματος. Το επόμενο όρισμα είναι ένα διάνυσμα που τερματίζεται σε NULL και περιέχει τα ορίσματα γραμμής εντολής σαν strings που τερματίζονται σε NULL.

Παρακάτω υπάρχουν παραδείγματα σύνταξης των εντολών execv() που χρησιμοποιούν αυτά τα ορίσματα:

char *const arg[]={“traverse”, “.”, “>”, “1000”, NULL};

char *const envp[]={“files=50”, NULL} ;

execv(“/path/traverse”, arg) ;

execve(“/path/traverse”, arg, envp) ;

execvp(“traverse”, arg) ;

Σε κάθε ένα από τα παραπάνω παραδείγματα κάθε execv() συνάρτηση δημιουργεί μια διεργασία που εκτελεί το πρόγραμμα traverse.

3. Καθορίζοντας περιορισμούς στις συναρτήσεις exec()

Υπάρχει ένα όριο στο μέγεθος των argv[] και envp[] που μπορούν να περαστούν στις συναρτήσεις exec(). Η συνάρτηση sysconf() μπορεί να χρησιμοποιηθεί για να καθοριστεί το μέγιστο μέγεθος των ορισμάτων γραμμής εντολής καθώς και το μέγεθος των περιβαλλοντικών παραμέτρων για τις exec() συναρτήσεις που δέχονται την παράμετρο envp[]. Για να επιστρέψει το μέγεθος, η name πρέπει να έχει την τιμή _SC_ARG_MAX.

#include <unistd.h>

long sysconf(int name);

Ένας άλλος περιορισμός όταν χρησιμοποιείται η exec () και οι άλλες συναρτήσεις που δημιουργούν διεργασίες είναι ο μέγιστος αριθμός των ταυτόχρονων διεργασιών που επιτρέπονται ανά ταυτότητα χρήστη. Για να επιστρέψει αυτό τον αριθμό η name πρέπει να έχει την τιμή _SC_CHILD_MAX.

4. Η συνάρτηση wait ()

Όπως αναφέρθηκε παραπάνω η κλήση συστήματος exec() φορτώνει ένα δυαδικό αρχείο στη μνήμη (καταστρέφοντας την εικόνα του προγράμματος που περιέχει την κλήση συστήματος exec() στη μνήμη) και ξεκινάει την εκτέλεσή του. Με αυτόν τον τρόπο, οι δύο διεργασίες (γονική και θυγατρική) είναι ικανές να επικοινωνήσουν και μετά να ακολουθήσουν διαφορετική πορεία. Η γονική διεργασία μπορεί στη συνέχεια να δημιουργήσει περισσότερες θυγατρικές, ή αν δεν έχει τίποτε άλλο να κάνει όσο οι θυγατρικές διεργασίες τρέχουν, μπορεί να καλέσει μία κλήση συστήματος wait() για να μεταφέρει τον εαυτό του έξω από την ουρά ετοιμότητας, μέχρι τον τερματισμό των θυγατρικών διεργασιών. Έτσι η γονική διεργασία περιμένει τη θυγατρική διεργασία να ολοκληρωθεί με την κλήση συστήματος wait(). Όταν η θυγατρική διεργασία ολοκληρωθεί (με έμμεση ή ρητή κλήση της exit()) η γονική διεργασία επανενεργοποιείται από την κλήση της wait(), ενώ ολοκληρώνεται χρησιμοποιώντας την κλήση συστήματος exit(). Κατά την ολοκλήρωσή της η θυγατρική διεργασία μπορεί να επιστρέψει μία τιμή κατάστασης (τυπικά έναν ακέραιο) στην γονική της διεργασία (μέσω της κλήσης συστήματος wait()). Όλοι οι πόροι της διεργασίας – συμπεριλαμβανομένων της φυσικής και εικονικής μνήμης, των ανοιχτών αρχείων και των buffers E/E – επανακτώνται από το λειτουργικό σύστημα. Όταν η γονική διεργασία πρέπει να περιμένει για την ολοκλήρωση της θυγατρικής της, τότε η παράμετρος της wait() είναι 0.

5. Διαβάζοντας και αναθέτοντας τιμές σε περιβαλλοντικές μεταβλητές

Οι περιβαλλοντικές μεταβλητές είναι strings που τερματίζονται με NULL και αποθηκεύουν πληροφορίες που εξαρτώνται από το σύστημα, όπως είναι τα μονοπάτια σε καταλόγους που περιέχουν εντολές, βιβλιοθήκες, και διαδικασίες που χρησιμοποιούνται από διεργασίες. Μπορούν επίσης να χρησιμοποιηθούν για να μεταδώσουν χρήσιμες πληροφορίες που ορίζονται από το χρήστη από τη γονική προς τη θυγατρική διεργασία. Παρέχουν ένα μηχανισμό για την παροχή συγκεκριμένων

πληροφοριών σε διεργασία χωρίς να χρειάζεται αυτές να περαστούν μέσα από τον κώδικα. Οι μεταβλητές αυτές είναι προκαθορισμένες και κοινές σε όλα τα κελύφη διεργασίες στο σύστημα αυτό. Οι μεταβλητές αυτές αρχικοποιούνται από τα αρχεία εκκίνησης. Παρακάτω φαίνονται μερικές κοινές μεταβλητές συστήματος:

$HOME	το απόλυτο μονοπάτι που οδηγεί στο βασικό κατάλογο
$PATH	μια λίστα από καταλόγους για την αναζήτηση εντολών
$MAIL	το απόλυτο μονοπάτι του γραμματοκιβωτίου
$USER	η ταυτότητα του χρήστη
$SHELL	το απόλυτο μονοπάτι του κελύφους
$TERM	ο τύπος του τερματικού

Μπορούν να αποθηκευτούν σε ένα αρχείο ή σε μια λίστα που θα περιέχει τις μεταβλητές περιβάλλοντος. Η λίστα αυτή θα περιέχει δείκτες σε strings που τερματίζονται σε NULL. Η μεταβλητή:

extern char *environ

δείχνει στην παραπάνω λίστα όταν η διεργασία αρχίσει να εκτελείται. Τα strings αυτά έχουν τη μορφή: name = value, όπως εξηγήθηκε νωρίτερα. Οι διεργασίες που αρχικοποιούνται με τις συναρτήσεις execl(), execlp(), execv(), execvp() κληρονομούν το περιβάλλον της γονικής διεργασίας. Οι διεργασίες που αρχικοποιούνται με τις συναρτήσεις execve() και execle() καθορίζουν το περιβάλλον της νέας διεργασίας.

Υπάρχουν συναρτήσεις και προγράμματα που μπορούν να κληθούν για να ερευνηθούν, να προστεθούν ή να τροποποιηθούν τέτοιες μεταβλητές. Η getenv() χρησιμοποιείται για να αποφασιστεί εάν μια συγκεκριμένη μεταβλητή έχει πάρει τιμή. Η παράμετρος name περιέχει το όνομα της μεταβλητής στην οποία γίνεται η αναφορά. Η συνάρτηση θα επιστρέψει NULL εάν η συγκεκριμένη μεταβλητή δεν έχει πάρει τιμή. Εάν η μεταβλητή έχει πάρει τιμή, η συνάρτηση θα επιστρέψει ένα δείκτη σε ένα string που περιέχει την τιμή. Δηλαδή, η σύνταξη των χρήσιμων εδώ συναρτήσεων είναι η παρακάτω:

#include <stdlib.h>

char *getenv(const char *name);

int setenv(const char *name, const char *value, int overwrite);

void unsetenv(const char *name);

Για παράδειγμα:

char *Path;

Path = getenv(“PATH”);

Το string Path παίρνει την τιμή που περιέχεται στην προκαθορισμένη μεταβλητή PATH. Η setenv() χρησιμοποιείται για να αλλάξει ή να προσθέσει μια μεταβλητή στο περιβάλλον της καλούσας διεργασίας. Η παράμετρος name περιέχει το όνομα της μεταβλητής που πρέπει να αλλάξει ή να πάρει τιμή. Παίρνει την τιμή που έχει η παράμετρος value. Εάν η παράμετρος overwrite έχει μη μηδενική τιμή τότε γίνεται αλλαγή τιμής. Αντίθετα, αν η overwrite έχει τιμή 0, το περιεχόμενο της μεταβλητής περιβάλλοντος δεν μεταβάλλεται. Η setenv() επιστρέφει 0 όταν ολοκληρωθεί

επιτυχώς και -1 διαφορετικά. Η unsetenv() διαγράφει την τιμή της μεταβλητής που περιέχεται στην παράμετρο name.

Χρησιμοποιώντας την system() για τη δημιουργία διεργασιών

Η συνάρτηση system() χρησιμοποιείται για να τρέξει μια εντολή ή ένα εκτελέσιμο πρόγραμμα. Η system() προκαλεί την εκτέλεση της δυάδας fork - exec και ενός κελύφους (shell). Η system() εκτελεί ένα fork και η θυγατρική διεργασία καλεί μία exec() με ένα κέλυφος που εκτελεί τη δοσμένη εντολή ή πρόγραμμα. Η σύνταξή της είναι η παρακάτω:

#include <stdlib.h>

int system(const char *string);

Η παράμετρος string μπορεί να είναι μια εντολή συστήματος, ή το όνομα ενός εκτελέσιμου αρχείου. Εάν ολοκληρωθεί επιτυχώς, η συνάρτηση επιστρέφει την κατάσταση τερματισμού της εντολής ή την τιμή του προγράμματος (αν το πρόγραμμα επιστρέφει κάποια). Σφάλματα μπορεί να συμβούν σε αρκετά επίπεδα, π.χ. η fork() ή η exec() μπορεί να παρουσιάσουν σφάλμα ή το κέλυφος μπορεί να μην μπορεί να εκτελέσει την συγκεκριμένη εντολή ή το πρόγραμμα.

Η συνάρτηση επιστρέφει μια τιμή στην γονική διεργασία. Η συνάρτηση επιστρέφει 127 εάν αποτύχει η exec() και -1 εάν συμβεί κάποιο άλλο σφάλμα.

Οι κλήσεις συστήματος exit (), kill () και abort ()

Υπάρχουν δύο συναρτήσεις που μία διεργασία μπορεί να καλέσει για να τερματίσει, οι exit () και abort (). Η exit () συνάρτηση προκαλεί τερματισμό της διεργασίας με κανονικό τρόπο. Όλα τα ανοικτά αρχεία που σχετίζονται με τη διεργασία κλείνουν. Η συνάρτηση πετά όλες τις ανοικτές ροές που περιέχουν δεδομένα που δεν έχουν αποθηκευτεί σε προσωρινό αποθηκευτικό χώρο, και κλείνει τις ροές αυτές. Η παράμετρος status αντιπροσωπεύει την κατάσταση εξόδου της διεργασίας. Επιστρέφεται στην γονική διεργασία που βρίσκεται σε αναμονή, η οποία και επανεκκινείται. Η τιμή της status μπορεί να είναι 0, EXIT_FAILURE, ή EXIT_SUCCESS. Η τιμή 0 σημαίνει ότι η διεργασία τερματίστηκε επιτυχώς. Η γονική διεργασία που βρίσκεται σε αναμονή έχει πρόσβαση όνο στα 8 λιγότερης σημαντικότητας bits της status. Εάν η γονική διεργασία δεν περιμένει για τη διεργασία να ολοκληρωθεί, η διεργασία αυτή υιοθετείται από την διεργασία init.

Η συνάρτηση abort () προκαλεί τον μη κανονικό τερματισμό της καλούσας διεργασίας. Ένας μη κανονικός τερματισμός της διεργασίας προκαλεί το ίδιο αποτέλεσμα με την fclose () σε όλες τις ανοικτές ροές. Η γονική διεργασία που βρίσκεται σε αναμονή θα λάβει ένα σήμα ότι η θυγατρική της διεργασίας τερματίστηκε μη κανονικά. Μια διεργασία πρέπει να τερματίζει με αυτόν τον τρόπο μόνο εφόσον συμβεί κάποιο λάθος το οποίο δεν μπορεί να αντιμετωπιστεί προγραμματιστικά.

#include <stdlib.h>

void exit (int status);

void abort (void);

Η συνάρτηση kill () μπορεί να χρησιμοποιηθεί για να προκαλέσει τον τερματισμό μιας άλλης διεργασίας. Η συνάρτηση kill () στέλνει ένα σήμα στη συγκεκριμένη διεργασία που έχει καθοριστεί ή κατέχει την παράμετρο pid. Η παράμετρος sig είναι το σήμα που στέλνεται στην καθορισμένη διεργασία. Τα σήματα βρίσκονται στο header file <signal.h>. Για να τερματιστεί μια διεργασία μέσω της kill (), η παράμετρος sig πρέπει να έχει την τιμή SIGKILL. Η καλούσα διεργασία πρέπει να έχει το απαραίτητο δικαίωμα για να στείλει ένα σήμα σε μία διεργασία, ή κατέχει ένα πραγματικό ή αποτελεσματικό user id που ταιριάζει με το πραγματικό ή το αποθηκευμένο user id της διεργασίας που λαμβάνει το σήμα. Η καλούσα διεργασία μπορεί να έχει το δικαίωμα να στέλνει μόνο συγκεκριμένα σήματα σε διεργασίες και όχι άλλα. Εάν η συνάρτηση αποστείλει το σήμα επιτυχώς, επιστρέφεται 0 στην καλούσα διεργασία. Εάν αποτύχει, επιστρέφεται -1.

Η καλούσα διεργασία μπορεί να στείλει το σήμα σε μία ή περισσότερες διεργασίες κάτω από τις ακόλουθες συνθήκες:

pid > 0 Το σήμα θα σταλεί στη διεργασία της οποίας το PID είναι ίσο με το pid.

pid = 0 Το σήμα θα σταλεί σε όλες τις διεργασίες των οποίων η ταυτότητα της ομάδας διεργασιών (group id) είναι ίδια με αυτή της καλούσας διεργασίας.

pid = -1 το σήμα θα σταλεί σε όλες τις διεργασίες για τις οποίες η καλούσα διεργασία έχει δικαίωμα να αποστείλει αυτό το σήμα.

pid < -1 Το σήμα θα αποσταλεί σε όλες τις διεργασίες των οποίων η ταυτότητα της ομάδας διεργασιών (group id) είναι ίση με την απόλυτη τιμή του pid και για την οποία η καλούσα διεργασία έχει δικαίωμα να αποστείλει το σήμα.

```
#include <signal.h>

int kill (pid_t pid, int sig);
```

Άσκηση

Να κατασκευάσετε ένα πρόγραμμα το οποίο δημιουργεί μια νέα διεργασία. Η θυγατρική διεργασία θα δημιουργεί ένα νέο κατάλογο και θα τοποθετεί εκεί ένα αρχείο που θα περιέχει το αποτέλεσμα της εκτέλεσης της εντολής ps -A. Θα εμφανίζει όλες τις ταυτότητες διεργασιών που ανήκουν στην ίδια ομάδα με αυτή.

Η γονική διεργασία θα κρυπτογραφεί το παραπάνω αρχείο, χρησιμοποιώντας ένα κλειδί από ένα άλλο αρχείο εισόδου και θα γράφει σε ένα τρίτο αρχείο την έξοδο της πράξης κρυπτογράφησης Τα τρία παραπάνω αρχεία θα δίνονται ως παράμετροι.

Η κρυπτογράφηση θα γίνει με χρήση του αλγόριθμου Vigenere. Ο αλγόριθμος αυτός εργάζεται ως εξής: τα γράμματα του αλφαβήτου κωδικοποιούνται με αριθμούς με αρχή το μηδέν. Δηλαδή το Α αντιστοιχεί στο 0, το Β στο 1, το C στο 2, κλπ. Ο αλγόριθμος παίρνει το κείμενο εισόδου και προσθέτει σε αυτό το κείμενο που αντιστοιχεί στο κλειδί, τοποθετώντας τα γράμματα το ένα κάτω από το άλλο και προσθέτοντας τα γράμματα που είναι στην ίδια θέση. Αν το κείμενο του κλειδιού είναι μικρότερο από αυτό του κειμένου εισόδου, τότε επαναλαμβάνεται. Για παράδειγμα παρακάτω κρυπτογραφείται το κείμενο SECURE με τη λέξη KEY για κλειδί:

S EC URE

KEY KEY

Η πράξη της κρυπτογράφησης είναι η εξής: c=(a+k)mod n, όπου a είναι ένα γράμμα του κειμένου εισόδου, k είναι το αντίστοιχο γράμμα κλειδιού, c το κρυπτογραφημένο γράμμα και n το σύνολο των γραμμάτων του αλφαβήτου. Οπότε στο παραπάνω παράδειγμα το κρυπτοκείμενο που θα παραχθεί θα είναι: CIAEVC.

Η υλοποίηση της άσκησης θα γίνει ως εξής: πρώτα θα διαβάζονται τα κείμενα του κλειδιού και του αρχικού κειμένου από τα αντίστοιχα αρχεία τους και αφού τοποθετηθούν σε πίνακες, θα ακολουθεί η πράξη κρυπτογράφησης. Οι πίνακες που θα χρησιμοποιηθούν πρέπει να είναι δυναμικοί και σε καμιά περίπτωση στατικοί. Για το λόγο αυτό θα χρησιμοποιηθούν οι γνωστές συναρτήσεις δέσμευσης μνήμης malloc και realloc.

Επίσης, το πρόγραμμα που θα κατασκευαστεί θα πρέπει να κάνει έλεγχο λαθών κατά τη σύνταξη της εντολής με κατάλληλη έκδοση διαγνωστικών μηνυμάτων.

Η διάρκεια της άσκησης είναι μία (1) εβδομάδα.