PROJECT 3

COMP211

Technical Univesrity of Crete

ODYSSEAS STAVROY 2018030199

Project 3:

Το 3ο project αποτελείτε από 2 προγράμματα γραμμένα σε C τα οποία καλύπτουν ένα αρκετά μεγάλο εύρος λειτουργιών. Συνοπτικά η άσκηση περιέχει μέσα, επικοινωνία με TCP/UDP σε διαφορετικά ports, pipes, πολλαπλούς file descriptors και έλεγχος αυτών, παραλληλισμό με διεργασίες "παιδιά" άλλα και signal handling/manipulation.

Το πρόγραμμα remoteServer:

Κατά το κάλεσμα του παίρνει 2 ορίσματα: το TCP port στο οποίο θα ακούει και τον αριθμό των παιδιών που θα δημιουργήσει.

Πατέρας:

- Αφου ξεκινήσει η κυρία διεργασία του προγράμματος(και μετά τη δήλωση των απαιτουμένων μεταβλητών) ο πατέρας δημιουργεί ένα pipe, αγνοεί το σήμα SIGPIPE και γεννάει τόσα παιδιά όσα δήλωσε ο χρήστης.
- Κάθε παιδί μόλις ξεκινήσει θέτει ως signal handler για το σήμα SIGUSR2 τη συνάρτηση suicide() και μεταβαίνει στη συνάρτηση child_server(), με όρισμα το read end του pipe, και μένει εκεί μέχρι να έρθουν τα κατάλληλα σήματα για τερματισμό της λειτουργίας του.
- Ο πατέρας συνεχίζει με το να δηλώνει ως signal handler του τη συνάρτηση kronos() για τα 2 σήματα SIGUSR1, SIGUSR2, κατασκευάζει το tcp socket με χρήση της συνάρτησης socket_init() και πλέον ακούει για τυχών συνδέσεις.
- Ο πατέρας ακολούθως αρχικοποιεί τα 2 sets με file descriptors που θα παρακολουθεί για πάντα με τη χρήση της select() και του FD_ISSET() και μπαίνει σε ένα ατέλειωτο βρόγχο για να ξεκινήσει την παρακολούθηση.
- Για κάθε file descriptor ελέγχει αν είναι μέσα στο set που υπάρχει δραστηριότητα, αν δεν είναι τότε προχωρά στην επομένη επανάληψη. Αν είναι υπάρχουν 2 σενάρια:
    - 1.) Είτε ο file descriptor είναι το tcp port μας, άρα κάποιος client προσπαθεί να συνδεθεί, έτσι κάνουμε accept και μας επιστρέφεται ένας καινούριος file descriptor τον οποίο προσθέτουμε στο set και συνεχίζουμε την επανάληψη.
    - 2.) Είτε ο file descriptor είναι ένας από τους πελάτες και μας έχει στείλει ένα μήνυμα. Τότε διαβάζουμε το μήνυμα του πελάτη σε ένα buffer και ελέγχουμε αν είναι έγκυρο, αλλιώς έχει αποσυνδεθεί και διαγράφουμε τον file descriptor του από τα sets. Ο πελάτης είναι γραμμένος με τέτοιο τρόπο ώστε πάντα οι

πρώτοι 10 χαρακτήρες του μηνύματος είναι το port στο οποίο ο πελάτης ακούει για udp συνδέσεις και μετά ακολουθεί μήνυμα. Στη συνεχεία ο πατέρας παίρνει τη διεύθυνση του πελάτη και καλεί τη συνάρτηση parse() και επιστρέφει πίσω μια καθαρή εντολή είτε ένα μήνυμα λάθους αν η εντολή είναι μη υποστηριζόμενη. Ο πατέρας αφού έχει αυτές τις 3 απαιτούμενες πληροφορίες τα γράφει όλα μαζί σε ένα άλλο buffer με αυτή τη μορφή:

20 bytes client's IP address | 10 bytes client's UDP port | 270 bytes msg

- Γράφει όλο το buffer στο pipe και αρχικοποιεί με μηδενικά όλα τα buffers που χρησιμοποιηθήκαν

- Επαναλαμβάνει συνεχώς αυτήν τη διεργασία γράφοντας συνεχεία νέες εντολές προς εκτέλεση για το παιδί.

Παιδί:

- Όλη η ζωή του παιδιού είναι μέσα στη συνάρτηση child_server()
- Με την είσοδο του σε αυτή τη συνάρτηση το παιδί αρχικοποιεί δικές του μεταβλητές που χρειάζεται και δημιουργεί ένα μισοτελειωμένο udp socket
- Το παιδί και αυτό με τη σειρά του μπαίνει σε ένα ατέλειωτο βρόγχο επανάληψης
- Διαβάζει συνεχώς σε ένα buffer με μέγεθος όσα γράφει ο πατέρας και σπάει το μήνυμα του πάτερα στα 3 χρήσιμα κομμάτια.
- Υπάρχουν 4ις βασικές εντολές που απασχολούν το παιδί:
  - Αν είναι (έγκυρη)εκτελέσιμη εντολή UNIX τότε με τη χρήση της popen() την τρέχει και κρατάει την έξοδο της σε ένα FILE *. Με αναγνώσεις των 512 bytes κάθε φορά στέλνει την έξοδο της στον πελάτη αφού πρώτα τροποποιεί τη διεύθυνση αποστολής/port του udp socket. Όταν τελειώσει στέλνει ένα μήνυμα "DONE" στον πελάτη που υποδηλώνει ότι η έξοδος της τρέχουσας εντολής εστάλην.
  - Αν η εντολή δεν είναι έγκυρη τότε στέλνει ένα μήνυμα στον πελάτη "bad" ώστε να κλείσει το αρχείο(κενό)
  - Αν η εντολή είναι "end" τότε το παιδί στέλνει στο σήμα SIGUSR1 στον πάτερα, κλείνει το read end του στο pipe και μεταβαίνει στη suicide().
  - Αν η εντολή είναι "timeToStop" τότε το παιδί στέλνει στο σήμα SIGUSR2 στον πάτερα, κλείνει το read end του στο pipe.

- Αρχικοποιεί με μηδενικά όλα τα buffers που χρησιμοποιηθήκαν.

bool valid():

- Επιστρέφει true αν η εντολή είναι έγκυρη αλλιώς false.
- Σημείωση: η διάσπαση μπορούσε να γίνει και με τη χρήση της strtok και ακολούθως η εκτέλεση με την exec.

Char * parse():

- Καθαρίζει την εντολή και επιστρέφει είτε "bad", όταν μη έγκυρη η εντολή/>100 χαρακτήρες, είτε την εντολή καθαρή.
- Σημείωση: καθαρίζει όλα τα spaces από την αρχή άλλα μετά τα " | " δεν επιτρέπετε να έχει πέραν του ενός space.

Void perror_exit():

- Τυπώνει το μήνυμα λάθους και τερματίζει το πρόγραμμα με EXIT_FAILURE

int socket_init():

- Με βάση τα ορίσματα δημιουργεί ένα tcp/udp socket και επιστρέφει τον file descriptor αυτού.

Void wait_for_ded():

- Καλείτε από τον πάτερα με το σήμα SIGUSR1, όταν ένα παιδί τερματίσει για να μαζέψει το zombie process που δημιουργείτε.

Void suicide():

- Καλείτε από ένα παιδί για να τερματίσει τη λειτουργία του όταν πάρει την εντολή "end" με τη χρήση του σήματος SIGUSR2

void kronos():

- Όταν ο πατέρας πάρει τα σήματα SIGUSR1/SIGUSR2 τότε μεταβαίνει στη συνάρτηση kronos() και ανάλογα με το σήμα εκτελεί την κατάλληλη λειτουργιά:
  - Αν το σήμα είναι SIGUSR1 τότε μεταβαίνει στη wait_for_ded() διότι ένα παιδί του τερμάτισε.

- Αν το σήμα είναι SIGUSR2 τότε ο πατέρας στέλνει το σήμα SIGUSR2 σε όλα τα παιδιά, κάνοντας τα να μεταβούν στη suicide() και περιμένει για ΟΛΑ τα παιδιά να τερματίσουν πριν να τερματίσει και αυτός.

Το πρόγραμμα remoteClient:

Το πρόγραμμα αυτό διαβάζει ένα αρχείο από εντολές και τις στέλνει μια-μια στον server. Έχει ως όρισμα το hostname του server, το port για tcp του server, το port που θα ακούει για udp και το αρχείο που περιέχονται οι εντολές.

- Όταν ξεκινήσει αρχικοποιεί τις απαιτούμενες μεταβλητές του και διαβάζει πόσες γραμμές έχει το αρχείο ΑΓΝΟΟΝΤΑΣ τις γραμμές με end/timeToStop.
- Δημιουργεί ένα παιδί που είναι υπεύθυνο μόνο για να ακούει udp συνδέσεις και να γράφει στα αρχεία εξόδου.
- Ο πατέρας είναι υπεύθυνος καθαρά για την αποστολή κάθε εντολής όπως έχει.

Πατέρας:

- Αφου δημιουργήσει το tcp socket του τότε καλεί τη συνάρτηση read_sent() με όρισμα τον FILE * και τον file descriptor του socket.
- Μέχρι να βρει το τέλος του αρχείου διαβάζει μια-μια γραμμή και τη στέλνει στον server.
- Αφου τελειώσει τότε περιμένει το παιδί να τερματίσει και αυτό.

Παιδί:

- Αφου και αυτό με τη σειρά του δημιουργήσει ένα udp socket μπαίνει σε ένα ατέλειωτο βρόγχο, ανοίγει το πρώτο αρχείο και περιμένει για πακέτα udp. Όταν φτάσει ένα αρχείο γράφει συνεχεία σε αυτό το αρχείο μέχρι να πάρει πακέτο “DONE”.
- Θα γράψει σε τόσα αρχεία όσα είναι η γραμμές που έχει διαβάσει στην αρχή το πρόγραμμα αγνοώντας τις εντολές end/timeToStop.
- Μόλις γράψει τον απαιτούμενο αριθμό αρχείων τερματίζει και συλλέγεται από τον πατέρα που με τη σειρά του τερματίζει και αυτός.

ΣΗΜΕΙΩΣΕΙΣ:

- Υπάρχουν αρκετά sleep() για να σιγουρέψω τον συγχρονισμό μεταξύ πάτερα και παιδιών κατά την εκκίνηση και τερματισμό τους.
- Το πρόγραμμα τυπώνει χρήσιμες πληροφορίες για την κατάσταση του(ποτέ συνδέθηκε κάποιος πελάτης, ποτέ αποσυνδέθηκε, τερματίζει κάποιο παιδί κτλ.)