NOM:LI

Prenom: Zheng

1.

1.1

1. Lorsqu'une socket en mode connecté est fermée, une lecture sur la socket à l'autre extrémité de la connexion renvoie 0 indiquant ainsi la fermeture de la communication (le read() ou recv() renvoie 0).

2. Son rôle est de créer un processus enfant à partir d'un processus existant, et le processus d'origine est appelé processus parent.
   Après avoir appelé fork (), lorsque le contrôle est transféré au code fork dans le noyau, le noyau commence à faire:
   a. 1. Allouez de nouveaux blocs de mémoire et de nouvelles structures de données du noyau aux processus enfants.
   b. 2. Copiez une partie de la structure de données du processus parent dans le processus enfant.
   c. 3. Ajoutez le processus enfant à la liste des processus système.
   d. 4. Fork revient pour démarrer le planificateur, la planification.
3. pipe est bidirectionnel

1.2

(1) Dans la phase de démarrage, lisez le fichier de configuration et créez un socket de type approprié (TCP ou UDP) pour chaque service dans le fichier de configuration. Chaque socket nouvellement créé est ajouté à un ensemble de mots de description qui seront utilisés par un appel de sélection.
(2) Appel d'appel (pour chaque socket. Ce numéro de port TCP ou UDP est obtenu par getservbyname. Les paramètres de la fonction sont le champ de nom de service et le champ de protocole du serveur correspondant dans le chapitre sur le fichier de configuration.
(3) Pour chaque socket TCP, appelez Listen pour accepter les demandes de connexion entrantes; cette étape n'est pas effectuée sur les sockets datagrammes.
(4) Après avoir créé toutes les sockets, appelez select et attendez qu'elles deviennent lisibles. La plupart de inetd est bloquée à l'intérieur de l'appel select, attendant qu'un socket devienne lisible.
(5) Lorsque select retourne qu'un socket est lisible, si le socket est un socket TCP et que le serveur est de type nowait, alors accept est appelé pour accepter la connexion.
(6) inetd appelle le processus dérivé de fork et le processus enfant gère la demande de service.

Le processus enfant ferme tous les descripteurs à l'exception du descripteur de socket à traiter (pour le socket renvoyé par accept pour TCP et le socket créé à l'origine pour UDP), le sous-processus appelle trois fois dup2 pour placer le descripteur du socket à traiter Copiez vers les mots de description 0, 1, 2, puis fermez l'ensemble d'origine des mots de description de l'interface. Par conséquent, les descripteurs ouverts par le sous-processus ne sont que 0, 1 et 2. Le processus enfant lit à partir de l'entrée standard, ce qui équivaut à

lire à partir du socket traité; le processus enfant écrit sur la sortie standard ou l'erreur standard, ce qui équivaut à écrire dans le socket traité.

Selon la valeur de configuration de login-name (user), le processus enfant appelle setgid et setuid pour se changer en utilisateur spécifié s'il n'est pas root.

Le processus enfant appelle exec pour exécuter le programme spécifié par le fichier de configuration pour traiter spécifiquement la demande.

(7) Si le socket TCP est renvoyé, le processus parent ferme d'abord le socket de connexion généré par la demande. Les appels de processus parent sélectionnent ici, en attendant que la prochaine socket devienne lisible.

1.3

-socket():Créer un socket

bind():Le socket créé pour la fonction socket () est associé à une adresse correspondante, et les données envoyées à cette adresse peuvent être lues et utilisées via le socket

listen():La fonction listen () marque sockfd comme un socket ouvert passivement et est utilisée comme paramètre d'acceptation pour recevoir les demandes de connexion entrantes.

-C’est serveur.

-Le multithreading convertit l'ordre des octets de l'hôte en ordre des octets du réseau et empêche les threads zombies

-Le processus d'appel n'a pas été interrompu

-Le socket n'est pas marqué comme fermé, puis revient immédiatement au processus d'appel

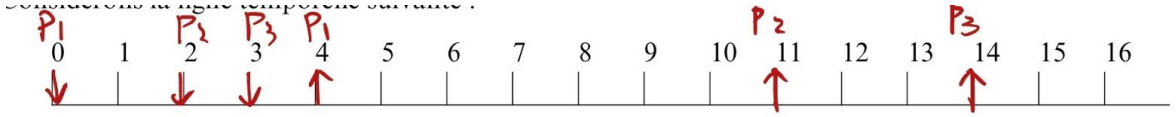
2

2.1

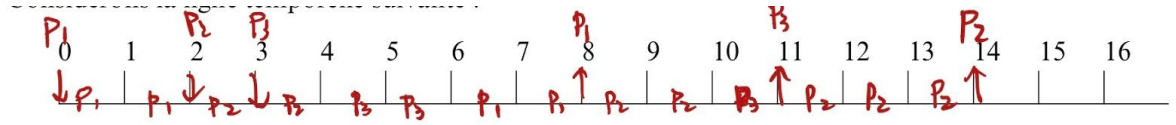
1.c
2.B
3.c
4.c
5.C
6.A
7.c
8.b
9.a
10.a
11.b
12.c
13.B
14.b
15.b

2.2

1.



2.



3. Fifo preemptif

2.3

La mémoire virtuelle est celle sur laquelle raisonne le programmeur M p
La mémoire virtuelle est une technologie de gestion de la mémoire du système informatique. Cela fait croire à l'application qu'elle a une mémoire disponible continue (un espace d'adressage continu et complet). En fait, elle est généralement divisée en plusieurs fragments de mémoire physique, et certains sont temporairement stockés sur un stockage sur disque externe. L'échange de données.
La mémoire physique est celle avec laquelle le processeur fonctionne M φ
Le système de mémoire virtuelle réalise donc une fonction φ M : M p ⟼ M φ
L'implémentation de la mémoire virtuelle doit être basée sur une gestion de la mémoire allouée discrètement