F2B205B - CO31A
Programmation Système et Réseaux
Partie Unix

C. Lohr

8 Décembre 2011

Modalités

- Documents de cours autorisés (polys, sujets de TP, listings de TP)
- Les équipements informatiques e ntout genre ne seront pas autorisés dans la salle. Ils devront tout au moins être éteints.
- Durée indicative : 1h30

1 The Daytime Protocol

On souhaite implémenter le service *Daytime*. Voila ce que dit Wikipedia [1] à ce sujet :

> *«The Daytime Protocol is a service in the Internet Protocol Suite, defined in 1983 in RFC 867. It is intended for testing and measurement purposes in computer networks.*
>
> *A host may connect to a server that supports the Daytime Protocol on either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) port 13. The server returns an ASCII character string of the current date and time in an unspecified format.»*

Le RFC 687 est donné en annexe A.

Sur la plupart des systèmes d'exploitation, ce bon vieux service est déclaré dans le fichier `/etc/services` de la manière suivante :

```
# Network services, Internet style
#
# Updated from http://www.iana.org/assignments/port-numbers and other
# sources like http://www.freebsd.org/cgi/cvsweb.cgi/src/etc/services

../..
daytime         13/tcp
daytime         13/udp
../..
```

Aussi, pour éviter de coder en dur la valeur de ce port udp, nous utiliserons la fonction standard `getservbyname()` qui parcourt ce fichier de configuration `/etc/services` et qui permet de retrouver le numéro de port d'un service en fonction de son nom et du protocole (tcp ou udp). L'annexe B documente cette fonction.

1. `http://en.wikipedia.org/wiki/Daytime_Protocol`

Les annexes C et D documentent des fonctions d'horloges utiles pour cet exercice.

On vous demande de réaliser un programme qui implémente ce service *daytime*. Nous nous restreindrons à un service en UDP et en IPv4. Nous ne gérerons pas l'aspect *daemon* (c.à.d. on fait un programme ordinaire que l'on lance en ligne de commande). Il n'est pas exigé de respecter scrupuleusement la syntaxe du langage C, mais tâchez de proposer quelque chose de ressemblant et compréhensible. À titre indicatif un tel programme tient en une page (et encore, avec les commentaires...).

2 Rétro ingénierie d'un code C

L'annexe E donne le code source d'un mystérieux programme. C'est une implémentation d'un vrai programme, d'une commande Unix standard. (Pour information c'est une version FreeBSD que j'ai simplifiée.)

Ce code ne comporte quasiment que des fonctions et appels systèmes standards vu en cours. (Excepté la fonction `isatty()` qui est documenté en annexe F.)

Listez les appels systèmes et mentionnez brièvement ce qu'ils font, d'une manière générale.
Expliquez ce qu'ils font dans le cadre de ce programme.
Puis, finalement expliquez en une phrase ou deux ce que fait ce programme.
Pourquoi est-ce qu'il n'y a aucun `EXIT_SUCCESS` dans ce programme ?
Question bonus : avez vous reconnu la commande Unix standard dont il est question ?
Seconde question bonus : ne trouvez vous rien de surprenant en début de ligne 40 ? Une idée ?

3 Questions de cours

- Quelles différences faites-vous entre des échanges via un *pipe* Unix et des échanges via une *socket* (p.ex. TCP) ?
- Comment créer des zombies ?
- `read() write() send() recv()` etc. sont des appels systèmes bloquants. Dans quel cas cela peut-il être gênant ? Comment contourner le problème (donnez 3 alternatives) ?

Annexe A

Network Working Group J. Postel
Request for Comments: 867 ISI
May 1983

Daytime Protocol

This RFC specifies a standard for the ARPA Internet community. Hosts on the ARPA Internet that choose to implement a Daytime Protocol are expected to adopt and implement this standard.

A useful debugging and measurement tool is a daytime service. A daytime service simply sends a the current date and time as a character string without regard to the input.

TCP Based Daytime Service

One daytime service is defined as a connection based application on TCP. A server listens for TCP connections on TCP port 13. Once a connection is established the current date and time is sent out the connection as a ascii character string (and any data received is thrown away). The service closes the connection after sending the quote.

UDP Based Daytime Service

Another daytime service service is defined as a datagram based application on UDP. A server listens for UDP datagrams on UDP port 13. When a datagram is received, an answering datagram is sent containing the current date and time as a ASCII character string (the data in the received datagram is ignored).

Daytime Syntax

There is no specific syntax for the daytime. It is recommended that it be limited to the ASCII printing characters, space, carriage return, and line feed. The daytime should be just one line.

Example:

Tuesday, February 22, 1982 17:37:43-PST

Annexe B

GETSERVENT(3) Manuel du programmeur Linux GETSERVENT(3)

NOM

getservbyname - Accéder aux informations sur les services

SYNOPSIS

```
#include <netdb.h>

struct servent *getservbyname(const char *name, const char *proto);
```

DESCRIPTION

La fonction getservbyname() renvoie une structure servent pour l'enregistrement du fichier /etc/services qui correspond au service nommé name et utilisant le protocole proto. Si proto est NULL, n'importe quel protocole sera accepté.

La structure servent est définie dans <netdb.h> ainsi :

```
struct servent {
    char  *s_name;       /* Nom officiel du service */
    char **s_aliases;    /* Liste d'alias */
    int    s_port;       /* Numéro de port */
    char  *s_proto;      /* Protocole à utiliser */
}
```

Les membres de la structure servent sont :

s_name Le nom officiel du service.

s_aliases
Une liste alias de noms de service terminée par NULL.

s_port Le numéro de port, donné dans l'ordre des octets du réseau.

s_proto
Le nom du protocole utilisé par ce service.

VALEUR RENVOYÉE

La fonction getservbyname() retourne un pointeur vers une structure servent statiquement allouée, ou un pointeur NULL si une erreur se produit, ou si la fin du fichier est atteinte.

VOIR AUSSI

getnetent(3), getprotoent(3), getservent_r(3), services(5)

Annexe C

TIME(2) Manuel du programmeur Linux TIME(2)

NOM
time - Lire l'heure

SYNOPSIS
#include <time.h>

time_t time(time_t *t);

DESCRIPTION
time() renvoie la date sous la forme du nombre de secondes depuis l'époque, 1er janvier 1970 à 00:00:00 (UTC).

Si t n'est pas NULL, la valeur renvoyée est également stockée dans la structure vers laquelle il pointe.

VALEUR RENVOYÉE
S'il réussit, l'appel time renvoie le nombre de secondes écoulées depuis l'époque. S'il échoue, la valeur ((time_t) -1) est renvoyée, et errno contient le code d'erreur.

ERREURS
EFAULT si t pointe en dehors de l'espace d'adressage.

VOIR AUSSI
date(1), gettimeofday(2), ctime(3), ftime(3), time(7)

Annexe D

CTIME(3) Manuel du programmeur Linux CTIME(3)

NOM

ctime - Convertir des dates et des temps au format ASCII

SYNOPSIS

```
#include <time.h>

char *ctime(const time_t *timep);
```

DESCRIPTION

La fonction ctime() prend un paramètre de type time_t qui représente un temps en seconde. Si l'on interprète ce paramètre comme une valeur absolue, il s'agit du nombre de secondes écoulées depuis l'époque, 1er janvier 1970 à 00:00:00 (UTC).

L'appel ctime(t) est équivalent à asctime(localtime(t)). Il convertit la date t en une chaîne de caractères, terminée par un caractère nul, de la forme

```
"Wed Jun 30 21:49:08 1993\n"
```

VALEUR RENVOYÉE

Cette fonction renvoie la valeur décrite ci-dessus, ou NULL si une erreur est détectée.

VOIR AUSSI

date(1), gettimeofday(2), time(2), utime(2), clock(3), difftime(3), strftime(3), strptime(3), timegm(3), tzset(3), time(7)

Annexe E

```
1  #include <sys/param.h>
2  #include <sys/stat.h>
3  #include <errno.h>
4  #include <fcntl.h>
5  #include <signal.h>
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <string.h>
9  #include <unistd.h>
10
11 #define FILENAME        "output.out"
12
13 void dofile() {
14   int fd;
15   char *p;
16
17   p = FILENAME;
18   if ((fd = open(p, O_RDWR | O_CREAT, S_IRUSR | S_IWUSR)) >= 0) {
19     if (dup2(fd, STDOUT_FILENO) == -1) {
20       perror(NULL);
21       exit(EXIT_FAILURE);
22     }
23     fprintf(stderr, "sending output to %s\n", p);
24   } else {
25     fprintf(stderr, "can't open a output.out file");
26     exit(EXIT_FAILURE);
27   }
28 }
29
30 int main(int argc, char *argv[]) {
31   if (argc < 2) {
32     fprintf(stderr, "usage: %s command\n", argv[0]);
33     exit(EXIT_FAILURE);
34   }
35
36   if (isatty(STDOUT_FILENO))
37     dofile();
38   if (isatty(STDERR_FILENO) && dup2(STDOUT_FILENO, STDERR_FILENO) == -1) {
39     /* may have just closed stderr */
40     fprintf(stdin, "%s: %s\n", argv[0], strerror(errno));
41     exit(EXIT_FAILURE);
42   }
43
44   signal(SIGHUP, SIG_IGN);
45   signal(SIGQUIT, SIG_IGN);
46
47   execvp(argv[1], &argv[1]);
48   fprintf(stderr, "%s", argv[1]);
49   exit(EXIT_FAILURE);
50 }
```

Annexe F

ISATTY(3) Manuel du programmeur Linux ISATTY(3)

NOM

isatty - Vérifier si un descripteur se rapporte à un terminal

SYNOPSIS

```
#include <unistd.h>

int isatty(int fd);
```

DESCRIPTION

isatty() teste si le descripteur de fichier ouvert fd fait référence à un terminal.

VALEUR RENVOYÉE

isatty() renvoie 1 si fd est un descripteur de fichier ouvert d'un terminal, sinon elle renvoie 0 et errno est définie.

ERREURS

EBADF fd n'est pas un descripteur de fichier valable.

EINVAL fd fait référence à un fichier à travers un terminal. POSIX.1-2001 spécifie l'erreur ENOTTY dans ce cas.