CSGE602055 Operating Systems
CSF2600505 Sistem Operasi
Minggu 08: Scheduling & Network Sockets Programming

Rahmat M. Samik-Ibrahim

Universitas Indonesia

http://rms46.vlsm.org/2/207.html

REV64 5-Sep-2017

Jadwal OS172

Minggu 00	29 Aug - 05 Sep 2017	Intro & Review
Minggu 01	07 Sep - 12 Sep 2017	IPR, REGEX, & Scripting
Minggu 02	14 Sep - 19 Sep 2017	Protection, Security, Privacy, & C-language
Minggu 03	26 Sep - 30 Sep 2017	BIOS, Loader, Systemd, & I/O
Minggu 04	03 Okt - 07 Okt 2017	Addressing, Shared Lib, Pointer & I/O Programming
Minggu 05	10 Okt - 14 Okt 2017	Virtual Memory
Ming. UTS	15 Okt - 24 Okt 2017	
Minggu 06	26 Okt - 31 Okt 2017	Concurency: Processes & Threads
Minggu 07	02 Nov - 07 Nov 2017	Synchronization
Minggu 08	09 Nov - 14 Nov 2017	Scheduling & Network Sockets Programming
Minggu 09	16 Nov - 21 Nov 2017	File System & Persistent Storage
Minggu 10	23 Nov - 28 Nov 2017	Special Topic: Blockchain
Cadangan	30 Nov - 09 Des 2017	
Ming. UAS	10 Des - 23 Des 2017	

Agenda

Week 08: Scheduling

- Reference: (OSCE2e ch6) (UCB 9/10) (UDA P3L1) (OLD 05)
- Scheduling
  - Basic Concepts
    - **WARNING:** It's just a BURST
    - IO Burst
    - CPU Burst
    - CPU Burst vs. Freq (OLD)
  - Utilization, throughput, {turnaround, waiting, response} time.
  - (Burst) Algorithm
    - FCFS
    - SJF
    - RR
    - Priority
    - Multilevel Queue
  - Preemptive / Non-preemptive Scheduling
  - I/O Bound / CPU Bound Processes
- Standard Linux Scheduling
  - Completely Fair Scheduler (CFS).
  - Real Time Scheduling.

Thread Scheduling

- Thread Scheduling
- Level
  - User-level thread scheduling
  - Kernel-level thread scheduling
- Contention Scope
  - Process-Contention Scope (PCS).
  - System-Contention Scope (SCS).
- Pthread
- MultiCore/ MultiProcessor/ MultiThread
  - affinity
  - load balancing
- Soft / Hard Real Time
- Big O Notation
  - O(1)
  - O(log N)
  - O(N)

Sockets

- Sockets
  - `atoi()`
  - `accept()`
  - `bind()`
  - `connect()`
  - `exit()`
  - `fprintf()`
  - `getenv()`
  - `gethostbyname()`
  - `htons()`
  - `listen()`
  - `memcpy()`
  - `memset()`

Sockets

- Sockets
  - `perror()`
  - `sizeof()`
  - `socket()`
  - `snprintf()`
  - `strchr()`
  - `strcmp()`
  - `strncpy()`
  - `strlen()`
  - `read()`
  - `write()`

server.c

```
/*
 * (c) 2007-2016 Rahmat M. Samik-Ibrahim -- This is free software
 * This program was copased from the net and hacked until it works.
 * Feel free to copy and/or modify and/or distribute it,
 * provided this notice, and the copyright notice, are preserved.
 * REV00 Tue Nov  8 11:45:35 WIB 2016
 * START Xxx Xxx XX XX:XX:XX UTC 2007
 */
char pesan[]="[FROM SERVER] ACK MESSAGE...\n";
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/socket.h>
#include <arpa/inet.h>
typedef struct sockaddr    sockad;
typedef struct sockaddr_in sockadin;
typedef struct hostent     shostent;
void error(char *msg){
   perror(msg);
   exit(0);
}
```

server 1

```
int main(int argc, char *argv[]) {
   char     buffer[256];
   int      clilen, newsockfd, nn, portno, sockfd;
   sockadin serv_addr, cli_addr;

   if (argc < 2) {
      fprintf(stderr, "ERROR, no port provided\n");
      exit(1);
   }
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
   if (sockfd < 0)
      error("ERROR opening socket");
   memset(&serv_addr, 0, sizeof(serv_addr));
   portno = atoi(argv[1]);
   serv_addr.sin_family      = AF_INET;
   serv_addr.sin_addr.s_addr = INADDR_ANY;
   serv_addr.sin_port        = htons(portno);
   if (bind(sockfd, (sockad*) &serv_addr, sizeof(serv_addr))< 0)
      error("ERROR on binding");
```

server 2

```
    listen(sockfd, 5);
    clilen = sizeof(cli_addr);
    newsockfd=accept(sockfd,(sockad*)&cli_addr,(socklen_t*)&clilen);
    if (newsockfd < 0)
       error("ERROR on accept");
    memset(buffer, 0, 256);
    nn = read(newsockfd,buffer,255);
    if (nn < 0)
       error("ERROR reading from socket");
    printf("[FROM CLIENT]:\n %s\n",buffer);
    nn = write(newsockfd, pesan, sizeof(pesan));
    if (nn < 0)
       error("ERROR writing to socket");
    return 0;
}
```

client.c

```
/*
 * (c) 2007-2016 Rahmat M. Samik-Ibrahim -- This is free software
 * This program was copased from the net and hacked until it works.
 * Feel free to copy and/or modify and/or distribute it,
 * provided this notice, and the copyright notice, are preserved.
 * REV00 Tue Nov  8 11:45:52 WIB 2016
 * START Xxx Xxx XX XX:XX:XX UTC 2007
 */
char pesan[]="[FROM SERVER] ACK MESSAGE...\n";
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/socket.h>
#include <arpa/inet.h>
typedef struct sockaddr    sockad;
typedef struct sockaddr_in sockadin;
typedef struct hostent     shostent;
void error(char *msg){
   perror(msg);
   exit(0);
}
```

client 1

```
int main(int argc, char *argv[]) {
   char      buffer[256];
   int       nn, portno, sockfd;
   sockadin  serv_addr;
   shostent* server;
   if (argc < 3) {
      fprintf(stderr, "usage %s hostname port\n", argv[0]);
      exit(0);
   }
   portno = atoi(argv[2]);
   sockfd = socket(AF_INET,SOCK_STREAM,0);
   if (sockfd < 0)
      error("ERROR opening socket");
   server = gethostbyname(argv[1]);
   if (server == NULL) {
     fprintf(stderr, "ERROR, no such host\n");
     exit(0);
   }
   memset(&serv_addr,0,sizeof(serv_addr));
   serv_addr.sin_family = AF_INET;
   memmove( &serv_addr.sin_addr.s_addr, server->h_addr, server->h_length);
   serv_addr.sin_port   = htons(portno);
```

client 2

```
    if(connect(sockfd,(const struct sockaddr*) &serv_addr, sizeof(serv_addr))<0)
        error("ERROR connecting");
    printf("Enter the message: ");
    memset(buffer,   0, 256);
    fgets (buffer, 255, stdin);
    nn = write(sockfd,buffer,strlen(buffer));
    if (nn < 0)
       error("ERROR writing to socket");
    memset(buffer, 0, 256);
    nn = read(sockfd,buffer,255);
    if (nn < 0)
       error("ERROR reading from socket");
    printf("%s\n",buffer);
    return 0;
}
```

Lab

- client
- server
- client-server

The End

- This is the end of the presentation.