

### 1> RPC Concepts & Definitions

What is RPC? Privilege separation?

What is stub code? What is marshalling?

What is server stub code? What is unmarshalling?

### 2> Implementing RPC in C. Simple stub code example:

### 3> How do you marshal an int? float? struct? Linked list? Graph?

### 4> What is IDL (Interface Design Language)?

### 5> Complexity and latency of RPC vs local calls?

### 6> Working with structured data

Transferring large amounts of structured data:

JSON vs xml vs Google Protocol Buffers

### 7> Challenge: What argument(s) to this program will cause it to print "Admin/Debug rights"?

```
#define N (20)
int admin, debug;
int histogram[N];

static int hash(char* str) {
    int c, h = 0; // sdbm hash
    while (c = *str++)
        h = c + (h << 6) + (h << 16) - h;
    return h;
}

int main(int argc, char**argv){

    while(argc>1) {
        char*word= argv[ --argc];
        int h = hash(word);
        histogram[ (h<0?-h:h) % N ] ++;
    }
    if(admin || debug) puts("Admin/Debug rights");
    return;
}
```

### > cp gotcha

What do these two lines do?

```
cp ../*.c .  
cp ../*.c
```

### > When would an anonymous mapping be useful?

```
void quit(char*mesg) {  
    fprintf(stderr,"%s\n",mesg);  
    exit(1);  
}  
  
void child(char* shared) {  
    for(int i = 0; i < 100;i++) {  
        sprintf(shared,"! The value of i is %d\n",i);  
        sleep(1);  
    }  
}  
  
void parent(char*shared) {  
    while(1) {  
        if(*shared) {  
            puts(shared);  
            *shared = 0;  
        }  
        sleep(1);  
    }  
}
```

```
int main() {
```

```
    // Use MAP_ANON instead of MAP_FILE  
    size_t size = 4096;
```

```
    char *shared_mem = mmap(NULL, size,  
        PROT_READ | PROT_WRITE,  
        MAP_ANON | MAP_SHARED, 0, 0);
```

```
    if (shared_mem == (char*)-1) quit("mmap  
failed");
```

```
    pid_t pid = fork();
```

```
    if(pid ==0) {  
        child(shared_mem);  
    } else {  
        parent(shared_mem);  
    }  
    exit(1);
```

```
}
```