部门 _____ 科室 _____ 姓名 _____ 工号 _____

1 改错题 (共计9题, 60分)

1.1 请找出下面代码中的隐患或者错误, 不需要改正。(4分)

```
BYTE *ComputeCheck(WORD16 dbHandle)
{
    WORD16 curRecNum, xorCheck;
    BYTE *p, p_check[6];
    WORD32 addCheck;

    xorCheck=0;
    addCheck=0;
    curRecNum=GetRecNum(dbHandle);
    if (curRecNum==0) return NULL;
    ......
    memcpy(p_check, &xorCheck, 2);
    memcpy(p_check+2, &addCheck, 4);
    return (BYTE *)&p_check;
}
```

1.2 请找出下面代码中的隐患或者错误, 说明故障原因并改正。(6分)

```
extern WORD16 GetSessionIdAndPUIFromRaw(BYTE *DataIn, PUI*ptPui);

void  DimDapProcessOtherMSG(T_InstanceData *ptInstance,
                            BYTE *DataIn,
                            T_DimCmdHdr *ptCmdHead,
                            WORD32 DataLength )
{
    T_DimCmdHdr *ptDimMsgHeader = ptCmdHead;
    PUI          tPUI;

    if(ptInstance == NULL)
    {
        return;
    }
    if(NULL == DataIn)
    {
        return;
    }
    /* 流程跟踪 */
#if (defined(DIM_PROCTRACE) && defined(DIM_ATCA_PSS))
    GetSessionIdAndPUIFromRaw((PCHAR)DataIn, &tPUI);
    if (DIM_PROC_MSG_REQ == ptDimMsgHeader->Flags_R)
    {
        wMsgType = DIM_PROC_MSGTYPE_REQ;
    }
    else
    {
        wMsgType = DIM_PROC_MSGTYPE_ACK;
    }
    DimProcProcessReportToOmc(ptDimMsgHeader->dwAppId,
                              &tPUI, wMsgType,
                              ptDimMsgHeader->CmdCode,
                              DIM_PROC_MSGDIRC_ACCEPT,
                                 UNUSED_FLAG_WORD32,
                              "DimDapProcessOtherMSG: DAP Reveive One
Message From SCTP/TCP");
#endif
}
```

1.3 请找出下面代码中的隐患或者错误, 说明故障原因并改正。(6分)

```
#define M_LocationAreaIdentification   0xb
#define MAX_CODE_LEN  1000
void Decode_LocationID(BYTE * pMsgIn)
{
    BYTE  byBytePositionIndex = 0;
    WORD16 wReturn = S_SUCCESS;
    if (pMsgIn == NULL)
    {
        return;
    }
    while(byBytePositionIndex < MAX_CODE_LEN && wReturn == S_SUCCESS)
    {
        if(*(pMsgIn + byBytePositionIndex) ==
            M_LocationAreaIdentification)
        {
            byBytePositionIndex++;
            wReturn = GetIE_LocationAreaIdentification(pMsgIn +
                byBytePositionIndex );
            byBytePositionIndex += 5;
        }
    }
}
```

1.4 请找出下面代码中的隐患或者错误, 说明故障原因并改正。(6分)

```
WORD32 C_SRV_StrLen(BYTE* pbStr, WORD16 wMaxStrLen)
{
    WORD32  dwLen = 0;
    if (pbStr == NULL)
    {
        dwLen = 0;
    }
    else
    {
```

```
        dwLen = strlen((CHAR *) pbStr);
        if (dwLen > wMaxStrLen)
        {
            dwLen = wMaxStrLen;
        }
    }
    return  dwLen;
}
void test()
{
    CHAR chBuf[ REASON_LTH];
    BYTE bTempLenth = 0;
    bTempLenth = C_SRV_StrLen(”Moved Temporarily”, REASON_LTH);
    memcpy(chBuf, (BYTE*)”Moved Temporarily”, bTempLenth);
    chBuf [ strlen(chBuf)+1]  = ’\0’;
    printf(”%s \n”,chBuf);
}
```

1.5 请找出下面代码中的隐患或者错误,说明故障原因并改正。(10分)

```
typedef struct tagTableStru{
    DB_HANDLE   hTable;
    char        lpTableName[ TABLE_NAME_LEN];
    BYTE        btFieldsNum;
    BOOL        bSave;
}TABLE_STRU, *LP_TABLE_STRU;

#define MAX_TABLE_NUM       100
TABLE_STRU  TableStru[ MAX_TABLE_NUM];

#define  MAKE_TBL_FILENAME(TableLoc, Dir, TblFileName, Postfix) \
    memset(TblFileName, 0, sizeof(TblFileName));                \
    strcat(TblFileName,TableStru[ TableLoc].lpTableName);       \
    strcat(TblFileName,”.”);                                    \
    strcat(TblFileName,Postfix)

extern STATUS   _Tables_Remove(LPSTR Dir, LPSTR Postfix)
{
    BYTE        FileName[ 80];
    DB_HANDLE   TableLoc = 0;
    FILE     *  dbFile;

    while (TableStru[ TableLoc].hTable != INVALID_DB_HANDLE)
    {
        if (TableStru[ TableLoc].bSave)
        {
            memset(FileName, 0, sizeof(FileName));
            MAKE_TBL_FILENAME(TableLoc, Dir, FileName, Postfix);
            if ((dbFile = fopen(FileName,”rb”)) != NULL)
            {
                fclose(dbFile);
                if (OK != remove(FileName))
                {
                    return ERROR;
                }
            }
            else
            {
                DbgMsg(INFORM_ERR,(”fail to open %s!\n”,FileName));
            }
            TableLoc++;
        }
    }
    return OK;
}
```

1.6 请找出下面代码中的隐患或者错误,说明故障原因并改正。(9分)

```
#define TEMP_BUF_LEN 64

WORD16 Syn_SetToDB(VOID *para_in )
{
    BYTE        location[ TEMP_BUF_LEN]   = {0};
    BYTE        location_2[ TEMP_BUF_LEN]     = {0};
    WORD32  *   pParaIn = NULL;
    WORD32      handle;
    WORD32      ret = 0;

    *pParaIn = *(WORD32 *)para_in;
    handle = Data_GetKey(ROOT , ROOT , strlen(ROOT));
    if(!Valid_Loc(handle))
    {
        ret = Data_Set( ROOT , ROOT, strlen(ROOT),
                        _DATA_OPRTYPE_NOWRITE, _DATA_TYPE_ENTRY,
                        NULL,0);
        if(ret != DATA_SET_SUC)
        {
            return ERR_SET_DB_FAIL ;
        }
    }
    snprintf(location,TEMP_BUF_LEN,”%s%s%s”,ROOT,NODE_SPLIT, SYN) ;
    handle = Data_GetKey(ROOT , location , strlen(location));
    if(!Valid_Loc(handle))
    {
        ret = Data_Set( ROOT , location, strlen(location),
                        _DATA_OPRTYPE_NOWRITE, _DATA_TYPE_BRANCH,
                           NULL, 0);
```

```
        if(ret != DATA_SET_SUC)
        {
            return ERR_SET_DB_FAIL ;
        }
    }
    snprintf(location_2,TEMP_BUF_LEN,"%s%s%s",location,NODE_SPLIT,
       SYN_ENABLE) ;
    handle = Data_GetKey(ROOT , location_2 , strlen(location_2));
    {
        ret = Data_Set( ROOT , location_2,strlen(location_2),
                        _DATA_OPRTYPE_NOWRITE ,_DATA_TYPE_WORD32,(
                           LPSTR)pParaIn,sizeof(WORD32));
        if(ret != DATA_SET_SUC)
        {
            return ERR_SET_DB_FAIL;
        }
    }
    return SUCC_AND_NOPARA;
}
```

1.7 请找出下面代码中的隐患或者错误,说明故障原因并改正。(6分)

```
DIM_RESULT DimGetConfigAvpByTree(BYTE *ptr, T_ConfigAvpGroup *
    ptConfigAvpGroup,WORD32 dwLen)
{
    ......
    TmpIndex = 0;
    dwUsedNum =ptConfigAvpGroup->dwUsedNum<=MAX_AVPCONFIG_NUM?
       ptConfigAvpGroup->dwUsedNum:MAX_AVPCONFIG_NUM;
    for(i = 0; i<dwUsedNum; i++)
    {
        ResultAvpNum = 0;
        dwAvpCode = ptConfigAvpGroup->tAvpConfig[i].AvpCode;
        dwAvpNum = ptConfigAvpGroup->tAvpConfig[i].AvpNum;
        for(j = 0;
            j<pParserGvar->tRawLocationRecord.dwRecordNum;
            j++)  /*在已经扫描过的 avp 中查找*/
        {
            if(dwAvpCode == pParserGvar->tRawLocationRecord.
               tRawLocation[j].dwAvpCode)
            {
                ptConfigAvpGroup->tAvpConfig[i].AvpLen[ResultAvpNum]
                   =
                    pParserGvar->tRawLocationRecord.tRawLocation[j].
                       dwAvpLen;
                if(++ResultAvpNum >= MAX_AVPGET_NUM)
                {
                    bGetAll = TRUE;
                    break;
                }
            }
        }
        if(ResultAvpNum >= dwAvpNum)  /*找全,跳出找下一个*/
        {
            ptConfigAvpGroup->tAvpConfig[i].ResultAvpNum =
               ResultAvpNum;
            bGetAll = TRUE;
            break;
        }
    }
    if(bGetAll)
    {
        bGetAll = FALSE;
        continue;
    }
    while(pCurrentRaw < pEndRaw)  /*在剩余码流中找*/
    {
        if(!NetBytesToAvpHeader((CHAR*)pCurrentRaw, &tAvpHead))
        {
            if (DIM_S_OK != DIM_PUB_SemaphoreV(ptDimSemaphore))
            {
                return ~DIM_S_OK;
            }
            return DIM_E_PAR_TRANS_ENDIAN_FAIL;
        }

        /*把扫描过的 avp 保存下来*/
        pParserGvar->tRawLocationRecord.tRawLocation[TmpIndex].
            dwAvpCode = tAvpHead.dwAvpCode;
    }
    ptConfigAvpGroup->tAvpConfig[i].ResultAvpNum = ResultAvpNum;
}

    ......
    return DIM_S_OK;
}
```

1.8 请找出下面代码中的隐患或者错误,说明故障原因并改正。(5分)

```
#define MAX_BLADE_NUM_PER_POOL   20

typedef
{
    BOOL8  blEnabled;
    BYTE   bConfigWeight;
    WORD16 wIp;
    ......
}T_BladeNode;
```

```
typedef struct
{
    BYTE        bAliveBladeNum;
    WORD16      wIndex;
    T_BladeNode atBladeArray[ MAX_BLADE_NUM_PER_POOL];
}T_BladePool;

/* 根据选定算法选取一个可用的 Blade */
int Mc_SlbChoose(T_VServer *pVServer, T_SlbInfo *pSlbInfo)
{
    T_BladePool *p = &(pVServer->tPool);
    WORD32      j  = 0;
    BYTE        cw = 0; /* 当前权值 */

    if( (0 == p->wNum) || (0 == p->bAliveBladeNum) )
    {
        return MCS_FAIL;
    }
    switch(pVServer->bSlbType)
    {
        /* 轮询,这种 SLB 方法需要记录上次的结果,包括 wIndex */
        case SLBTYPE_ROUNDROBIN:
        {
            j = p->wIndex;
            do
            {
                j = (j+1) % MAX_BLADE_NUM_PER_POOL;
                if( p->atBladeArray[j].blEnabled )
                {
                    p->wIndex = j;
                    pSlbInfo->bBladeIndex = j;
                    pSlbInfo->wBIp        = p->atBladeArray[j].wIp;
                    return MCS_OK;
                }
            }while(j != p->wIndex);
            return MCS_FAIL;
        }

        /* 根据权重进行轮询,这种 SLB 方法需要使用记录上次的结果,
         * 包括CurWeight, wIndex 这种方式是先设置 CurWeigh
         * 为最大值,然后从头到尾寻找 Weight 大于等于此值的, Blade
         * 降低 CurWeight 为 CurWeight-gcd(S),然后再从头到尾寻找
         * Weight 大于等于此值的 Balde */
        case SLBTYPE_WEIGHTROUNDROBIN:
        {
            j  = p->wIndex;      /* 获取上次使用 Blade Index */
            cw = p->bCurWeight; /* 获取当前使用的 Weight */
            while(1)
            {
                j = (j+1) % MAX_BLADE_NUM_PER_POOL;
                if(0 == j) /* 表示 Pool 中第一个, Blade
                            * 初始时,或者循环了一圈再次开始 */
                {
                    if(cw <= p->bWeightGCD)
                    {
                        cw = p->bWeightMax;
                        if(0 == cw)
                        {
                            return MCS_FAIL;
                        }
                    }
                    else
                    {
                        cw -= p->bWeightGCD;
                    }
                }
                if(  p->atBladeArray[j].blEnabled &&
                     p->atBladeArray[j].bConfigWeight >= cw)
                { /* 选定此 Blade */
                    pSlbInfo->bBladeIndex = j;
                    pSlbInfo->wBIp = p->atBladeArray[j].wIp;
                    p->wIndex = j; /*记录下当前使用的 Blade index */
                    p->bCurWeight  = cw; /*记录下当前使用的 CurWeight */
                    return MCS_OK;
                }
            }
        }
        default:
            return MCS_FAIL;
    }
}
```

1.9 请找出下面代码中的隐患或者错误,说明故障原因并改正。(8分)

```
int sotcpbind(struct inpcb_tcp *pcb, struct sockaddr_in *addr)
{
    struct tcb *tcb = 0;
    ipaddr_t  laddr_t;
    UINT32 laddr = 0;
    UINT16  lport=0;
    struct brs_socket *so= 0;
    int wild = 0;
    int reuseport = 0;

    lport  = 0;
    tcb = pcb->tcpcb;
    so = tcp_get_socket(tcb);
    reuseport = (so->so_options & SO_OPT_REUSEPORT);
```

```
/* TCB 完整性检查 */
if (!tcb || tcb->state != TCP_CLOSED || tcb->rcv_buf == 0)
{
    return BRS_SOCKET_ERROR;
}

/* 只有在没有 reuse 选项打开的时候,查找 tcb 的时候才允许通配*/
if((so->so_options & (SO_OPT_REUSEADDR| SO_OPT_REUSEPORT) )==0)
    wild = LOOKUP_WILDCARD;
if(addr)  /* 显式调用 */
{
    /* 获取指定的地址和端口 */
    SET_IPV4_IPADDR(&laddr_t, &(addr->sin_addr.s_addr));
    SET_IPADDR_TYPE((&laddr_t), IPV4);  /* 一定是 ipv4 */
    lport = addr->sin_port;
    laddr = addr->sin_addr.s_addr;
    /* 检查是否组播地址 */
    if(ipaddr_test(&laddr_t, IPADDR_TEST_MULTICAST))
    {
        if(so->so_options & SO_OPT_REUSEADDR)
            reuseport = (SO_OPT_REUSEADDR | SO_OPT_REUSEPORT);
    }
    /*指定了特定地址,非通配地址*/
    else if(!ipaddr_test(&laddr_t, IPADDR_TEST_UNSPECIFIED))
    {
        /* 检查该地址是否是本地地址 */
#if !INSTALL_VOIP
            /* 检查该地址是否是本地地址或者 vrrp 地址*/
#if INSTALL_ALG && INSTALL_ATTACHE_VRRP
            if(IsMyAddr(&laddr_t) == 0 && IsVrrpAddr(&laddr_t) == 0 )
#else
                if(IsMyAddr(&laddr_t) == 0)
#endif
                {
                    return ESO_ADDRNOTAVAIL;
                }
#endif
        }
        /* 检查指定地址,端口是否有冲突 */
        if(lport)
        {
            struct tcb *t = tcp_tcb_lookup(laddr, lport, 0, 0,wild);
            if (t)
            {
                /* 找到了本地地址,端口冲突的 tcb */
                struct brs_socket *s = (struct brs_socket *)
                    tcp_get_socket(t);
                if((reuseport & s->so_options)==0)
                { /* 没有设置 reuseport 选项 */
                    return ESO_ADDRINUSE;
                }
            }
        }
        /* 保存本地地址*/
        tcp_set_local_address(tcb, laddr);
    }
    /* 将 tcb 按照本地地址,端口作索引,插入到 hash 表中 */
    add_tcp_tcb_to_hash(tcb);
    return BRS_SOCKET_OK;
}
```

2 填空题 (共计5题, 每题4分, 共20分)

2.1 以下程序的运行结果是 _____

```
char strBuff[8] = "1234567";
strncpy(strBuff, "abcd", 3);
printf("%s",strBuff);
```

2.2 以下程序的运行结果是 _____

```
main(int argc, char** argv)
{
    char *str="hello world";
    printf("%d,%d,%d\n",
           sizeof(str),
           sizeof("hello world"),
           strlen(str));
}
```

2.3 以下程序的运行结果是 _____

```
void GetMemory(char **p, int num)
{
    *p=(char *) malloc(num);
}
void main(void)
{
    char *str=NULL;
    GetMemory(&str,100);
    strcpy(str,"hello world");
    printf(str);
}
```

2.4 以下程序的运行结果是 _____

```
char *GetMemory(void)
{
    char p[]="hello world";
    return p;
}
void Test(void)
{
    char *str=NULL;
    str = GetMemory( );
    printf(str);
}
```

2.5 以下程序的运行结果是 _____

```
void GetMemory(char *p, int num)
{
    p=(char *) malloc(num);
}
void main(void)
{
    char *str=NULL;
    GetMemory(str,100);
    strcpy(str,"hello world");
    printf(str);
}
```

3 编程题 (共计2题, 每题10分, 共计20分)

3.1 链表排序(从小到大), 10分

节点定义为：

```
struct Node
{
    int nValue;
    struct Node* pNext;
};
```

最后一个节点的pNext = NULL。
函数原型：Node* SortChain(Node* pHead); pHead为指向链表头节点的指针。返回值：链表头

3.2 设有一个表头指针为pHead的单链表，节点定义同题目1。试设计一个算法，通过遍历一趟链表，将链表中所有结点的链接方向逆转，要求逆转结果链表的表头指针pHead指向原链表的最后一个结点。10分

函数原型：`Node* SortChain( Node* pHead );` pHead为指向链表头节点的指针。返回值：链表头