LISTA 5 – MAB353 2020-2 Remoto - 95 pontos - v0

Q5.1) (30 pontos) Se precisar referenciar algum bit de um valor inteiro de 32 bits assuma o formato <b31, b30,…,b1,b0>, com b31 sendo o bit mais significativo (MSB) e b0 sendo o bit menos significativo (LSB). Os códigos de montagem nesta questão não precisam ser comentados linha a linha, mas utilizados para responder ao que é indagado nos itens da questão.

a) O seguinte código C imprime o valor de retorno de um término normal do processo filho:

```
int   main() {
      int status; fork(); wait(&status);
      if (WIFEXITED(status)) printf("%d \n", WEXITSTATUS(status));
      exit(1); }
```

O código de montagem referente ao C, que não precisa ser comentado a priori, segue:

```
main:
1     endbr32
2     leal      4(%esp), %ecx
3     andl $-16, %esp
4     pushl     -4(%ecx)
5     pushl     %ebp
6     movl      %esp, %ebp
7     pushl     %ecx
8     subl      $20, %esp
9     call  fork
10    subl      $12, %esp
11    leal      -12(%ebp), %eax
12    pushl     %eax
13    call  wait
14    movl      -12(%ebp), %eax
15    addl $16, %esp
16    testb     $127, %al
17    je    .L4
.L2:
18    subl      $12, %esp
19    pushl     $2
20    call  exit
.L4:
21    subl      $4, %esp
22    movzbl    %ah, %eax
23    pushl     %eax
24    pushl     $.LC0
25    pushl     $1
26    call  __printf_chk
27    addl $16, %esp
28    jmp .L2
```

a1) (5) Para testar que houve um término normal do processo filho, alguns bits da variável status são testados. Quais são estes bits e qual o estado dos bits que indica que houve um término normal do processo filho? Não adianta apenas responder sem justificar. É preciso que você mostre as linhas pertinentes no código de montagem que justificam sua resposta.

a2) (5) Onde o valor de término do processo filho fica armazenado? Justifique pelo código de montagem. É preciso que você explique as linhas pertinentes no código de montagem que justificam a sua resposta.

a3) (2) Quais os valores possíveis de término de um processo filho, em geral? No exemplo, o filho retornará 1. Queremos saber qual os valores possíveis para retorno. Justifique e explique usando as linhas pertinentes do código de montagem.

b) Para imprimir o sinal que causou uma parada do processo filho, temos o seguinte código C :

```
int   main() {
      int status; fork(); wait(&status);
      if (WIFSTOPPED(status)) printf("%d \n", WSTOPSIG(status));
      exit(1); }
```

A parte relevante do código de montagem gerado é :

```
      …
1     call  fork
2     subl      $12, %esp
3     leal      -12(%ebp), %eax
4     pushl     %eax
5     call  wait
6     movl      -12(%ebp), %eax
7     addl $16, %esp
8     cmpb      $127, %al
9     je    .L4
.L2:
10    subl      $12, %esp
11    pushl     $2
12    call  exit
.L4:
13    subl      $4, %esp
14    movzbl    %ah, %eax
15    pushl     %eax
16    pushl     $.LC0
17    pushl     $1
18    call  __printf_chk
19    addl $16, %esp
20    jmp .L2
```

b1) (4) Para identificar o sinal que causou a parada do processo filho alguns bits da variável status são testados. Quais são estes bits e qual o estado destes bits indica o processo filho está parado? Não adianta apenas responder sem justificar. É preciso que você mostre as linhas pertinentes no código de montagem que justificam sua resposta.

b2) (4) Onde o valor do sinal que causou a parada está armazenado e quais os valores possíveis deste sinal? Justifique e explique usando as linhas pertinentes do código de montagem.

c) Para imprimir o sinal que interrompeu o processo filho, temos o seguinte código C :

```
int   main() {
      int status; fork(); wait(&status);
      if (WIFSIGNALED(status)) printf("%d \n", WTERMSIG(status));
      exit(1); }
```

A parte relevante do código de montagem gerado é :

```
      ...
1     call  fork
2     subl      $12, %esp
3     leal  -   12(%ebp), %eax
4     pushl     %eax
```

```
5    call  wait
6    movl      -12(%ebp), %edx
7    movl      %edx, %eax
8    andl $127, %eax
9    addl $1, %eax
10   addl $16, %esp
11   cmpb      $1, %al
12   jg    .L4
.L2:
13   subl      $12, %esp
14   pushl     $2
15   call  exit
.L4:
16   subl      $4, %esp
17   andl $127, %edx
18   pushl     %edx
19   pushl     $.LC0
20   pushl     $1
21   call  __printf_chk
22   addl $16, %esp
23   jmp .L2
```

c1) (5) Para identificar que o processo filho foi interrompido por um sinal, alguns bits da variável status são testados. Quais são estes bits e qual o estado destes bits indica o processo filho sofreu uma interrupção? Não adianta apenas responder sem justificar. É preciso que você mostre as linhas pertinentes no código de montagem que justificam sua resposta.

c2) (5) Onde o valor do sinal que interrompeu está armazenado e quais os valores possíveis deste sinal? Justifique e explique usando as linhas pertinentes do código de montagem.

Q5.2) (20 pontos) Considere o seguinte código C:

```
void end (void) {printf("2");}
int main(){
      if (fork()!=0) atexit (end);
      if (fork()==0) printf("0"); else {wait(); printf("1");}
      exit(0);}
```

A função atexit recebe como argumento um ponteiro para uma função. Esta função é acrescentada a uma lista de funções (inicialmente vazia) que serão executadas quando a função exit é chamada. Assuma que o sistema operacional executará a rotina *end* imediatamente com o término do processo, fazendo com que o 2 apareça sempre imediatamente após a impressão provocada por main. Apresente e justifique todas as saídas possíveis deste programa. Lembre-se que processos podem ser executados em qualquer ordem.

Q5.3) (20 pontos) Liste o sistema operacional em uso. Rode o programa abaixo e copie a tela do terminal com a impressão de saída. Você pode precisar usar alguns destes includes: sys/types.h, sys/wait.h, stdio.h, stdlib.h, signal.h, errno.h.

```
int main() {
      if (fork()==0) {printf(“a”); exit(0);}
      else {sleep(5); printf(“b”); waitpid(-1,NULL,0);}
      printf(“c”);
      exit(0);}
```

a) Justifique a saída do programa em termos das chamadas de sistema utilizadas.

Agora, rode o programa modificado abaixo e copie a tela do terminal com a impressão de saída.

```
int main() {
      if (fork()==0) {sleep(5); printf(“a”); exit(0);}
      else {printf(“b”); waitpid(-1,NULL,0);}
      printf(“c”);
      exit(0);}
```

b) Justifique a saída do programa em termos das chamadas de sistema utilizadas. Encontre justificativa plausível. O programa deve ser executado exatamente como no enunciado, sem qualquer alteração, e sua saída analisada e explicada.

Q5.4) (25 pontos) Para contar o número de vezes que um sinal é enviado ao processo pai, o seguinte código C é criado. Para compilar, pode ser necessário usar alguns destes includes no seu sistema: sys/types.h, unistd.h, stdio.h, signal.h, errno.h, setjmp.h, wait.h, stdlib.h.

```
int n = 0;
int pid;
void conta(int sig) {
   n++;
   sleep(1);
   return;
   }

int main() {
  int i;
  signal(SIGUSR1,conta);
  pid = fork();
  if (pid==0) {
     for (i=0; i< 5; i++) {
        kill(getppid(),SIGUSR1);
        printf ("enviado SIGUSR1 ao pai \n");
        }
     exit(0);
     }
   wait(NULL);
   printf("n=%d\n", n );
   exit(0);
  }
```

A saída esperada do programa é que sejam impressas 5 linhas “enviado SIGUSR1 ao pai” seguidas de uma linha “n= 5”.

a) (5) Capture numa mesma tela de terminal a compilação e execução do programa. Analise o programa, explique a interação entre os processos e justifique a saída obtida.

b) (5) Elimine o comando sleep(1) da rotina conta e recompile. Rode algumas vezes e se houver diferentes saídas, captura a tela que demonstra isso.

c) Para eliminar o problema de corrida entre os processos pai e filho é sugerida a seguinte modificação no programa:

```
int n = 0;
int pid;
void conta(int sig) {
   n++;
   kill(pid, SIGCONT);
   return;
   }

int main() {
  int i;
  signal(SIGUSR1,conta);
  pid = fork();
  if (pid==0) {
     for (i=0; i< 5; i++) {
        kill(getppid(),SIGUSR1);
        printf ("enviado SIGUSR1 ao pai \n");
        kill(getpid(), SIGSTOP);
        }
     exit(0);
     }
   wait(NULL);
   printf("n=%d\n", n );
   exit(0);
}
```

c1) (5) Explique a ideia do programador ao fazer a modificação no código. Isso vai funcionar na sua opinião? Justifique.

c2) (5) Capture numa mesma tela de terminal a compilação e execução do programa. Era o que seria esperado? Analise o programa, explique a interação entre os processos e justifique a saída obtida.

c3) (5) Qual a modificação no código deste item c que você faria para obter a saída correta se pudesse acrescentar apenas um sleep(1) no código acima? Justifique seu raciocínio para a solução proposta, faça a alteração e comprove se funciona, capturando a tela do terminal.