

Lab. di Sistemi Operativi

Esercitazioni proposte per le lezioni del 29-30/04/10

Utilizzando il compilatore gcc in Linux e disponendosi in gruppi di due persone per ogni PC del laboratorio.

1. Utilizzo della funzione write. Creare un sorgente in linguaggio c (es61.c) che riversi su stdout la stringa costante "Ciao, mondo!".
2. Modificare es61.c in modo che i caratteri siano scritti su stderr. Verificare il comportamento del programma così modificato utilizzando la ridirezione della shell.
3. Visualizzare la man page di write. Attenzione: esistono differenti man page per write, visualizzare quella nella sezione di nostro interesse (2, syscall).
4. Modificare es61.c in modo che il valore ritornato da write venga reso disponibile come exit value del processo.
5. Modificare es61.c in modo che il file descriptor venga assegnato dal primo argomento della linea di comando, o venga imposto pari a 1 se assente. Cosa avviene se si indica un fd diverso da 1 o 2?
6. Modificare es61.c in modo che la stringa scritta contenga il valore numerico di fd. Utilizzare un array di char e popolarlo con sprintf.
7. Differenze fra printf e write. Modificare es61.c in modo che l'emissione del carattere di fine linea avvenga con qualche secondo di ritardo ed osservare il risultato. Provare lo stesso con la printf.
8. Creazione di file. Si utilizzi la funzione open() in un sorgente c (es62.c) il cui obiettivo sia quello di creare un file nella directory corrente avente per nome la stringa rappresentata dal primo argomento.
9. Modifichiamo es62.c in modo da recuperare il valore di ritorno della open ed utilizzarlo come file descriptor per una write.
10. Chiusura di un file: la funzione close. Modificare es62.c in modo che il file creato sia chiuso prima della terminazione.
11. Utilizzo di un file esistente. Progettare un programma c (es63.c) che apra il file indicato dal primo argomento e sovrascriva i primi 4 caratteri con la stringa "ciao"
12. Cosa succede se il file vittima è più lungo di 4 caratteri? E se fosse più corto?
13. Modificare es63.c in modo che il file venga troncato alla dimensione 4.
14. Lettura da file. Creare un sorgente c (es64.c) che legga i primi N caratteri ($N < 100$) dal file F con N ed F rispettivamente primo e secondo argomento della command line.
15. Modificare es64.c in modo che gli N caratteri letti siano copiati nello stesso file a partire dalla posizione K, con K terzo argomento.

Soluzione

Esercitazioni proposte per le lezioni del 29-30/04/10

Utilizzando il compilatore gcc in Linux e disponendosi in gruppi di due persone per ogni PC del laboratorio.

1. Utilizzo della funzione write. Creare un sorgente in linguaggio c (es61.c) che riversi su stdout la stringa costante "Ciao, mondo!".

Soluzione:

```
/* file: es61.c
 * job: scrittura su stdout con write
 */
#include <unistd.h>
int main(int argc, char **argv) {
    write(1, "Ciao, \_\_mondo!\n", 13);
    return(0);
}
```

2. Modificare es61.c in modo che i caratteri siano scritti su stderr. Verificare il comportamento del programma così modificato utilizzando la ridirezione della shell.

Soluzione:

```
/* file: es61.c
 * job: scrittura su stdout con write
 */
#include <unistd.h>
int main(int argc, char **argv) {
    write(2, "Ciao, \_\_mondo!\n", 13);
    return(0);
}
/* $ ./es61 > /dev/null
 * $ Ciao, mondo!
 * $ ./es61 2> /dev/null
 * $
 */
```

3. isualizzare la man page di write. Attenzione: esistono differenti man page per write, visualizzare quella nella sezione di nostro interesse (2, syscall).

Soluzione:

```
$ man 2 write
```

4. Modificare es61.c in modo che il valore ritornato da write venga reso disponibile come exit value del processo.

Soluzione:

```
/* file: es61.c
 * job: scrittura su stdout con write
 */
#include <unistd.h>
```

```

int main(int argc, char **argv) {
    /* l'exit value del processo coincide
     * con il valore di ritorno della write
     * ovvero con il numero di caratteri
     * effettivamente scritti. */
    return(write(2,"Ciao , _mondo!\n" ,13));
}

```

5. Modificare es61.c in modo che il file descriptor venga assegnato dal primo argomento della linea di comando, o venga imposto pari a 1 se assente. Cosa avviene se si indica un fd diverso da 1 o 2?

Soluzione:

```

/* file: es61.c
 * job: scrittura su stdout con write
 */
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char **argv) {
    int fd;
    if (argc<2) {
        /* nessun argomento */
        fd=1;
    } else {
        /* fd in argv[1], numero */
        fd=atoi(argv[1]);
    }
    return(write(fd,"Ciao , _mondo!\n" ,13));
}

```

6. Modificare es61.c in modo che la stringa scritta contenga il valore numerico di fd. Utilizzare un array di char e popolarlo con sprintf.

Soluzione:

```

/* file: es61.c
 * job: scrittura su stdout con write
 */
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

char stringa[256];
int main(int argc, char **argv) {
    int fd;
    if (argc<2) {
        fd=1;
    } else {
        fd=atoi(argv[1]);
    }
    sprintf(stringa,256,"fd=%d: _Ciao , _mondo!\n" ,fd);
    return(write(fd ,stringa ,strlen(stringa)));
}

```

7. Differenze fra printf e write. Modificare es61.c in modo che l'emissione del carattere di fine linea avvenga con qualche secondo di ritardo ed osservare il risultato. Provare lo stesso con la printf.

Soluzione:

```
/* file: es61.c
 * job: write e printf, un po' di prove
 */
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

char stringa[256];
int main(int argc, char **argv) {
    int fd;
    if (argc < 2) {
        fd=1;
    } else {
        fd=atoi(argv[1]);
    }
    /* rimosso il fine linea '\n' */
    snprintf(stringa, 256, "fd=%d: _Ciao, _mondo!", fd);
    write(fd, stringa, strlen(stringa));
    sleep(3); /* attendo 3 secondi */
    write(fd, "\n", 1);
    /* subito una printf: cosa succede? */
    printf("fd=%d: _Ciao, _mondo!", fd);
    sleep(3); /* attendo altri 3 secondi */
    printf("\n"); /* solo ora, dopo i 3 secondi,
                   la printf mostra il suo effetto:
                   i caratteri della prima printf
                   sono stati ritardati fino al \n
                   */
    return(0);
}
```

8. Creazione di file. Si utilizzi la funzione open() in un sorgente c (es62.c) il cui obiettivo sia quello di creare un file nella directory corrente avente per nome la stringa rappresentata dal primo argomento.

Soluzione:

```
/* file: es62.c
 * job: open
 */
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main(int argc, char **argv) {
    if (argc < 2) {
        /* manca il nome, usciamo */
        write(2, "Inserire _il _nome _del _file!\n", 28);
        exit(1);
    }
}
```

```

    } else {
        open(argv[1], O_WRONLY|O_CREAT|O_EXCL, 00666);
        /* attenzione: la costante ottale 00666,
         * relativa ai permessi, viene applicata
         * al file dopo la mascheratura ad opera
         * di umask */
    }
    return(0);
}

```

9. Modifichiamo es62.c in modo da recuperare il valore di ritorno della open ed utilizzarlo come file descriptor per una write.

Soluzione:

```

/* file: es62.c
 * job: open
 */
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <stdio.h>

char msg[256];
int main(int argc, char **argv) {
    int fd;
    if (argc < 2) {
        /* manca il nome, usciamo */
        write(2, "Inserire il nome del file!\n", 28);
        exit(1);
    } else {
        fd=open(argv[1], O_WRONLY|O_CREAT|O_EXCL, 00666);
        snprintf(msg, sizeof(msg), "fd=%d\n", fd);
        write(1, msg, strlen(msg));
    }
    /* utilizziamo la write su fd: */
    return(write(fd, "Ciao, Mondo!\n", 13));
    /* fd e' un nuovo file descriptor che puo' essere
     * utilizzato esattamente come stdout o stderr */
}

```

10. Chiusura di un file: la funzione close. Modificare es62.c in modo che il file creato sia chiuso prima della terminazione.

Soluzione:

```

/* file: es62.c
 * job: open, close
 */
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>

```

```

#include <fcntl.h>
#include <string.h>
#include <stdio.h>

char msg[256];
int main(int argc, char **argv) {
    int fd;
    int nw; /* numero di caratteri scritti */
    if (argc < 2) {
        /* manca il nome, usciamo */
        write(2, "Inserire il nome del file!\n", 28);
        exit(1);
    } else {
        fd=open(argv[1], O_WRONLY|O_CREAT|O_EXCL, 00666);
        snprintf(msg, sizeof(msg), "fd=%d\n", fd);
        write(1, msg, strlen(msg));
    }
    nw=write(fd, "Ciao, Mondo!\n", 13);
    if (close(fd) != 0) {
        write(2, "Errore in chiusura\n", 20);
    }
    return(nw);
}

```

11. Utilizzo di un file esistente. Progettare un programma c (es63.c) che apra il file indicato dal primo argomento e sovrascriva i primi 4 caratteri con la stringa "ciao"

Soluzione:

```

/* file: es63.c
 * job: modifica di file
 */
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <stdio.h>

char msg[256];
int main(int argc, char **argv) {
    int fd;
    int nw; /* numero di caratteri scritti */
    if (argc < 2) {
        /* manca il nome, usciamo */
        write(2, "Inserire il nome del file!\n", 28);
        exit(1);
    } else {
        fd=open(argv[1], O_WRONLY);
        snprintf(msg, sizeof(msg), "fd=%d\n", fd);
        write(1, msg, strlen(msg));
        if (fd < 0) {
            snprintf(msg, sizeof(msg), "Errore in apertura di %s", argv[1]);
            write(2, msg, strlen(msg));
        }
    }
}

```

```

        exit(1);
    }
}
nw=write(fd,"ciao",4);
if(close(fd)!=0) {
    write(2,"Errore_in_chiusura\n",20);
}
return(nw);
}

```

12. Cosa succede se il file vittima è più lungo di 4 caratteri? E se fosse più corto?

Soluzione:

La dimensione finale del file e' il massimo fra 4 ed il numero di caratteri precedente

13. Modificare es63.c in modo che il file venga troncato alla dimensione 4.

Soluzione:

```

/* file: es63.c
 * job: modifica di file
 */
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <stdio.h>

char msg[256];
int main(int argc, char **argv) {
    int fd;
    int nw; /* numero di caratteri scritti */
    if (argc < 2) {
        /* manca il nome, usciamo */
        write(2,"Inserire_il_nome_del_file!\n",28);
        exit(1);
    } else {
        fd=open(argv[1],O_WRONLY);
        /* |O_TRUNC azzera la lunghezza del file all'atto
         * della open */
        snprintf(msg,sizeof(msg),"fd=%d\n",fd);
        write(1,msg,strlen(msg));
        if (fd < 0) {
            snprintf(msg,sizeof(msg),
                "Errore_in_apertura_di_%s",argv[1]);
            write(2,msg,strlen(msg));
            exit(1);
        }
    }
    nw=write(fd,"ciao",4);
    ftruncate(fd,4); /* troncamento manuale */
    if(close(fd)!=0) {
        write(2,"Errore_in_chiusura\n",20);
    }
}

```

```

    }
    return(nw);
}

```

14. Lettura da file. Creare un sorgente c (es64.c) che legga i primi N caratteri (N<100) dal file F con N ed F rispettivamente primo e secondo argomento della command line.

Soluzione:

```

/* file: es64.c
 * job: lettura di N caratteri da F
 * uso: es64 N F
 */
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

char array[101]; /* 100 + '\0' */
char msg[256]; /* messaggio */
int fd; /* file descriptor */
int N; /* primo arg */
char *F; /* secondo arg */
int nr; /* numero char letti */

int main(int argc, char **argv) {
    switch (argc) { /* num args */
        case 3:
            N=atoi(argv[1]);
            F=argv[2];
            break;
        default:
            snprintf(msg, sizeof(msg),
                "uso: %s \u00d9N\u00d9F\u00d9n", argv[0]);
            write(2, msg, strlen(msg));
            exit(1);
    }
    fd=open(F, O_RDONLY);
    if (fd<0) { /* chk open */
        snprintf(msg, sizeof(msg),
            "Errore \u00d9in \u00d9apertura\u00d9n");
        write(2, msg, strlen(msg));
        exit(1);
    }
    if (N>100) { /* chk N */
        snprintf(msg, sizeof(msg),
            "N \u00d9troppo \u00d9grande\u00d9n");
        write(2, msg, strlen(msg));
        exit(1);
    }
    nr=read(fd, array, N);
    if (nr<0) { /* chk read */

```

```

        snprintf(msg, sizeof(msg),
        "Errore in lettura\n");
        write(2, msg, strlen(msg));
        exit(1);
    }
    array[nr]='\0'; /* termino la stringa */
    snprintf(msg, sizeof(msg),
        "letto: \n%s\nTotale %d caratteri\n", array, nr);
    write(1, msg, strlen(msg));
    return(0);
}

```

15. Modificare es64.c in modo che gli N caratteri letti siano copiati nello stesso file a partire dalla posizione K, con K terzo argomento.

Soluzione:

```

/* file: es64.c
 * job: lettura di N caratteri da F
 * e copia in posizione K
 * uso: es64 N F K
 */
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

char array[101]; /* 100 + '\0' */
char msg[256]; /* messaggio */
int fdr; /* file descriptor, lettura */
int fdw; /* file descriptor, scrittura */
int N; /* primo arg */
char *F; /* secondo arg */
int K; /* terzo arg */
int nr; /* numero char letti */
int nw; /* numero char scritti */

int main(int argc, char **argv) {
    switch (argc) { /* num args */
        case 4:
            N=atoi(argv[1]);
            F=argv[2];
            K=atoi(argv[3]);
            break;
        default:
            snprintf(msg, sizeof(msg),
                "uso: %s N F K\n", argv[0]);
            write(2, msg, strlen(msg));
            exit(1);
    }
    fdr=open(F, O_RDONLY);
    if (fdr < 0) { /* chk open */

```

```

        snprintf(msg, sizeof(msg),
        "Errore_in_apertura_(rd)\n");
        write(2,msg, strlen(msg));
        exit(1);
    }
    if (N>100) { /* chk N */
        snprintf(msg, sizeof(msg),
        "N_troppo_grande\n");
        write(2,msg, strlen(msg));
        exit(1);
    }
    nr=read(fdr , array ,N);
    if (nr<0) { /* chk read */
        snprintf(msg, sizeof(msg),
        "Errore_in_lettura\n");
        write(2,msg, strlen(msg));
        exit(1);
    }
    fdw=open(F,O_WRONLY);
    if (fdw<0) { /* chk open */
        snprintf(msg, sizeof(msg),
        "Errore_in_apertura_(wr)\n");
        write(2,msg, strlen(msg));
        exit(1);
    }
    if (lseek(fdw,K,SEEK_SET)!=K) {
        /* retval=posizione */
        snprintf(msg, sizeof(msg),
        "Errore_nel_posizionamento\n");
        write(2,msg, strlen(msg));
        exit(1);
    }
    nw=write(fdw , array , nr);
    snprintf(msg, sizeof(msg),
        "Letti_%d_caratteri\n"
        "Copiati_%d_caratteri_alla_posizione_%d\n",
        nr,nw,K);
    write(1,msg, strlen(msg));
    return(0);
}

```