5 Potoki

5.1 Potoki nienazwane

1. Utwórz potok funkcją systemową `pipe`, zapisz do niego przykładowy napis, a następnie odczytaj:

```
#include <stdio.h>
#include <unistd.h>

int main()
{
  int fd[2];
  char buf[100];

  pipe(fd);
  write(fd[1], "Hello", 6);
  read(fd[0], buf, 6);
  printf("%s\n", buf);
  return 0;
}
```

2. Przenieś proces zapisu do potoku do procesu potomnego. Proces macierzysty powinien utworzyć potok, a następnie uruchomić proces potomny:

```
pipe(fd);
if (fork()==0)
{
  write(...);
  exit(0);
}
...
```

   Dodając oczekiwanie funkcją `sleep` w procesie potomnym przed zapisem do potoku można zaobserwować pracę dwóch procesów w systemie. Proces macierzysty (czytający) czeka na dane z potoku.

3. Napisz program, który zrealizuje przetwarzanie potokowe: `ls | tr a-z A-Z`.

```
#include <stdio.h>
#include <unistd.h>
```

```
int main()
{
  int fd[2];

  pipe(fd);
  if (fork()==0)
  {
    dup2(fd[1], 1);
    execlp("ls", "ls", NULL);
  }
  else {
    dup2(fd[0], 0);
    execlp("tr", "tr", "a-z", "A-Z", NULL);
  }
  return 0;
}
```

Uzupełnij kod programu tak, aby się poprawnie kończył.

4. Uzupełnij program z poprzedniego punktu tak, aby wynik przetwarzania trafiał do pliku `out.txt`, jak w poniższym zleceniu:

```
# ls | tr a-z A-Z > out.txt
```

5. Napisz program, który wykona następujące zlecenie:

```
# ls -l /tmp | sort -n +4 | tail -5
```

6. Napisz funkcję, która zwraca pełną domenową nazwę komputera. Nazwa powinna być pobierana z polecenia:

```
# hostname -f
```

5.2 Potoki nazwane

1. Napisz dwa programy, które skomunikują się za pośrednictwem łącza nazwanego. Pierwszy program powinien utworzyć takie łącze i zapisać do niego wybrany łańcuch tekstowy, a drugi powinien z tego łącza odczytać dane. Przykładowa implementacja programu piszącego:

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

int main()
{
  int fd;

  mkfifo("fifo", 0600);
  fd = open("fifo", O_WRONLY);
  write(fd, "Hello", 6);
  close(fd);
  return 0;
}
```

2. Zrealizuj zlecenie `ps ax | tr a-z A-Z`, wykorzystując do komunikacji potok nazwany.

5.3 Komunikacja dwukierunkowa

1. Napisz program komunikujący się dwukierunkowo ze swoim procesem potomnym. Proces potomny powinien dopisać do przekazanego łańcucha tekstowego napis „OK" i przesłać go do procesu macierzystego. Należy skorzystać z pary lokalnych gniazdek komunikacyjnych tworzonych funkcją systemową `socketpair`:

```
int fd[2];

if ((socketpair(PF_UNIX, SOCK_STREAM, 0, fd))==-1)
{
  perror("socketpair");
  exit(1);
}
```