Operating systems 1, Lecture 3

Sanda-Maria AVRAM, Ph. D.

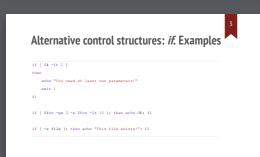
```
The grammar of shell (cont.)

command:

basicCommand [Production to the continuation of the continuation of the continuation of the command.ist [Production to the continuation of the con
```

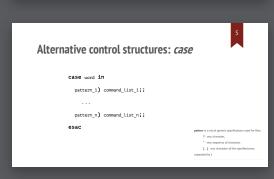
```
Alternative control structures: if

if command_list_1
then command_list_2
elif command_list_3
then command_list_4
...
elif command_list_n
then command_list_n+1
else command_list_n+2
fi
```



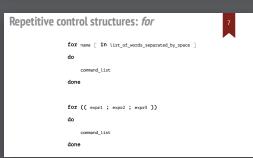
```
Alternative control structures: if. Examples

| an ascerak ("") will expand to literally anything
| if [[ "Sattingwar" == "sallting" ]]; then ...
| if [[ "Sattingwar" == "isallting" ]]; then ...
| combining conditional expressions using "sis" and "]|"
| if [[ Soun == 0] &s "Sattingwar" == foo ]]; then
| (()) may contain arithmetic operators, such as "==", "c" and ">=".
| can combine expressions with "sis" and "||" (but not with -a and -o )
| if (( Soun <= 3 )); then ...
```



```
Alternative control structures: case. Examples

case $1 in
    [n=2][(n-2]) echo "letter";;
    [0-3]) echo "digit";
    *) echo "no letter, nor digit";
esac
```



```
Repetitive control structures: for. Examples

for x in one two three foury do each number $xy done

for fis in 'is.'y do cat $fisy done

s=0;
for i in 'seq 1 10'; do s='expr $s + $i'; done

factorial=1

N=15

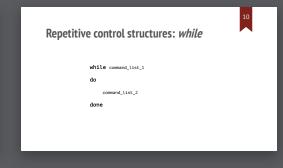
for ((i=2; $i<=Ny i++ ))

do

factorial=$((factorial * $i ))

done
echo $factorial
```







```
Repetitive control structures: while

while read a
do
case Se in
[s=11[(n-1)] echo "letter";;
[0-91) echo "digit";
) echo "m letter, nor digit"
exit;;

desac
done
```



```
Repetitive control structures: until

count=1
until [-e *8** ]
do
ech "This is parameter number Scount"
shift
count='expr Scount + 1'
done

factorial=1; [Si=13; Si=2;
until [6] = yt SN ]
do
factorial=0((Sfactorial * Si ))
i='expr Si + 1'
done
```

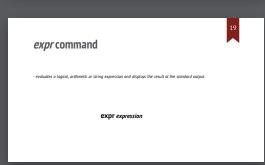


```
continue, break. Examples

for ((//))
do read Var
if [ "%var" -"." ]; then break; fi
done

for fin "$8"
do if (-f %(f).bak | # if .bak backup file exists, read next file
then
eacho "Mateping &f file..."
continue # read ment file and skip op command
fi
# here means no backup file exists, just use op command to copy file
//kin/op &f &f.bak
done
```





```
expr command with logical expressions

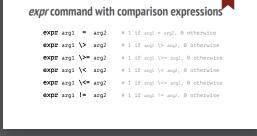
expr arg1 \& arg2 # arg1 if arg1 and arg2 != 0 or NULL, 0 otherwise

expr arg1 \| arg2 # arg1 if arg1 and arg2 != 0 or NULL, arg2 otherwise
```

```
expr command with arithmetic expressions
with integers

expr arg1 \* arg2  # = arg1 * arg2
expr arg1 / arg2  # = arg1 / arg2
expr arg1 \* arg2  # = arg1 / arg2
expr arg1 \* arg2  # = arg1 * arg2
expr arg1 \* arg2  # = arg1 + arg2
expr arg1 - arg2  # = arg1 - arg2
```







```
expr command with string expressions

expr string: expr <-> expr match string regexpr

# number of characters in string identical to those in expr

expr length string # number of characters in string

expr substr string pos len

# substring of string which starts at pos and has length len

expr index string chars

# the smallest position of a character in chars in string
```



```
expr command with string expressions.

Examples (cont.)

* expr length "aba"

* expr substr "abcdefg" 1 2

ab

* expr index "abcdefg" dac

* expr index "abcdefg" dac

* expr index "abcdefg" dc

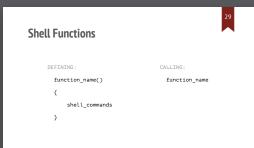
* expr index "abcdefg" dc

* expr index "abcdefg" dc

1

* expr index "abcdefg" dc

1
```





The End