3

COMMANDS

Dieses Kapitel behandelt Folgendes:

- ☞ Alternativer Befehlszugriff
- ☞ Befehlsprotokollierung
- ☞ Querverweise (Cross References)
- ☞ Fehlernachrichten
- ☞ Interaktive Befehlsverarbeitung
- ☞ Menüs
- ☞ Parameter
- ☞ Bedienerführung
- ☞ Shortcut-Commands
- ☞ UIM
- ☞ Mit Commands arbeiten

Die AS/400 ist berüchtigt für ihre intuitive Befehlsschnittstelle. Wenn Sie wissen, welchen Zweck ein Befehl erfüllen soll – wie z.B. das Konfigurieren von TCP-IP-Diensten – können Sie leicht den richtigen Namen des Befehls erraten (CFGTCP in diesem Fall). Intuitive Befehlsnamen sind einfach eines der vielen Dinge, die die AS/400 zu einem benutzerfreundlichen System machen. Darüber hinaus wurde in die AS/400 die Funktionalität integriert, die Sie benötigen, um Ihre eigene intuitive Commands zu erstellen, mit der Sie neue Funktionen ausführen können. Sie haben also ein System, das in Bezug auf die einfache Verwendung und entwicklerfreundliche Schnittstelle nicht zu schlagen ist.

Die Erstellung von Commands kann wie alles andere auf der AS/400 einen Schwierigkeitsgrad zwischen „sehr einfach“ und „sehr schwierig“ haben. Dabei hängt sehr viel vom gewünschten Komplexitätsgrad ab. Aus diesem Grund kann die Arbeit mit OS/400-Commands und die Erstellung Ihrer eigenen benutzerdefinierten Commands manchmal zu einer schwierigen Aufgabe werden. Und deshalb werden Sie davon profitieren, wenn Sie von Menschen unterstützt werden, die bereits Erfahrungen gesammelt haben und sich erfolgreich mit bestimmten Bereichen dieser komplexen Aufgaben befasst haben. Daher ist dieses Kapitel zu Commands für Sie wichtig.

In diesem Kapitel finden Sie einige der besten Tipps und Tricks zu Commands, die im Laufe der Jahre im *Midrange Computing* Magazin veröffentlicht wurden. Finden Sie heraus, wie Sie Ihre eigenen Abkürzungen für AS/400-Commands erstellen können, wie Ad-hoc-Commands erstellt werden können, wie mit Befehlsparametern gearbeitet wird und vieles mehr.

Shannon O’Donnell

Alternativer Befehlszugriff

Verwendung von Systemanforderungen für die Ausführung eines Commands

Die AS/400 erlaubt Ihnen das Versenden einer Nachricht in eine Nachrichtenwarteschlange durch Verwendung der Taste für die Systemabfrage mit Option 5 (Nachricht senden). Dies kann auch dann geschehen, wenn die Eingabe auf Ihrer Arbeitsstation eingeschränkt wurde (anders als mit dem Abruftasten-Programm). Durch Verwendung des in Abbildung 3.1 gezeigten Programms kann die Option „Nachricht senden“ verwendet werden, um einen Befehl zu senden, der durch das Betriebssystem ausgeführt wird.

Das Programm EXCMD (Abbildung 3.1) führt einen Befehl aus, der an das Programm von einer Nachrichtenwarteschlange aus gesendet wird. Nach Erstellung des Programms geben Sie die in Abbildung 3.2 gezeigten Commands ein oder fügen diese Ihrem Anfangsprogramm hinzu.


```
/* Folgender Befehl muss ausgeführt werden, bevor EXCMD verwendet +
        werden kann: +
        ADDLIBLE LIB(QTEMP) +
        CRTMSGQ MSGQ(QTEMP/Q) +
        CHGMSGQ MSGQ(QTEMP/Q) DLVRY(*BREAK) PGM(EXCMD) */

  EXCMD: +
     PGM PARM(&QUE &LIB &KEY)
```

Abbildung 3.1: Der Befehl EXCMD kann das Senden von Commands auch bei eingeschränkten Arbeitsstationen vereinfachen (Teil 1 von 2).

```
DCL VAR(&QUE) TYPE(*CHAR) LEN(10)
DCL VAR(&LIB) TYPE(*CHAR) LEN(10)
DCL VAR(&KEY) TYPE(*CHAR) LEN(4)
DCL VAR(&CMD) TYPE(*CHAR) LEN(132)

MONMSG MSGID(CPF0000)

RCVMSG MSGQ(&LIB/&QUE) KEYVAR(&KEY) MSG(&CMD)
CALL PGM(QCMDEXC) PARM(&CMD 132)

ENDPGM
```

Abbildung 3.1: Der Befehl EXCMD kann das Senden von Commands auch bei eingeschränkten Arbeitsstationen vereinfachen (Teil 2 von 2).

```
ADDLIBLE    LIB(QTEMP)
CRTMSGQ     MSGQ(QTEMP/Q)
CHGMSGQ     MSGQ(QTEMP/Q) DLVRY(*BREAK) PGM(EXCMD)
```

Abbildung 3.2: So wird mit dem Befehl EXCMD gearbeitet.

Um zu einem beliebigen Zeitpunkt einen Befehl einzugeben, drücken Sie die Taste für die Systemabfrage und geben „5“ gefolgt von einem Befehl in einfachen Anführungszeichen sowie den Parameter TOMSGQ(Q) ein. Zum Beispiel:

```
5 ,wrksbmjob *job‘ tomsgq(q)
```

Wenn Sie eine Bedienerführung für den Befehl wünschen, setzen Sie vor den Befehl ein Fragezeichen (z.B. ?WRKSBMJOB). Sie können diese Technik auch verwenden, um mit CALL QCMD zu prüfen, ob Fehlernachrichten generiert wurden.

Robin Klima

Befehlsprotokollierung

Protokollierung aller CL-Commands

Frage: Kennen Sie ein Tool, das die Aktionen der angemeldeten Anwender während des Tages speichern, anzeigen oder drucken kann? Ich weiß, die Protokollhistorie zeigt das Datum und den Zeitpunkt, zu dem sich der Anwender an der AS/400 angemeldet hat, und ob Nachrichten vorhanden sind. Ich würde jedoch auch gerne die verschiedenen Jobs sehen, die diese ausgeführt haben (interaktive und Stapeljobs).

Antwort: In V2R3 und späteren Versionen können Sie die Benutzerprofil-Protokollierung aktivieren, die das System veranlasst, für jeden Befehl, den ein bestimmter Benutzer eingibt, einen Journaleintrag zu erstellen. Anschließend können Sie diese Journaleinträge anzeigen, um herauszufinden, welche Commands von einem Anwender ausgeführt wurden. Gehen Sie wie folgt vor, um dieses Feature zu aktivieren:

1. Prüfen Sie, ob Bibliothek QSYS ein Journal mit der Bezeichnung QAUDJRN enthält. Wenn ja, wechseln Sie zu Schritt 4.

```
CHKOBJ OBJ(QSYS/QAUDJRN) OBJTYPE(*JRN)
```

2. Erstellen Sie ein Journalempfangsprogramm. Geben Sie dem Programm einen beliebigen gültigen Namen und speichern Sie es in einer beliebigen Bibliothek.

```
CRTJRNRCV JRNRCV(xxx/journal-receiver)
```

3. Erstellen Sie das System-Protokollierungsjournal. Das Journal muss als QAUDJRN bezeichnet werden und muss in Bibliothek QSYS gestellt werden. Geben Sie im Parameter JRNRCV das Journalempfangsprogramm an, das Sie in Schritt 2 erstellt haben.

```
CRTJRN JRN(QSYS/QAUDJRN) JRNRCV(xxx/journal-receiver)
```

4. Ändern Sie den Protokollierungs-Steuersystemwert, um die Protokollierung zu aktivieren.

```
CHGSYSVAL SYSVAL(QAUDCTL) VALUE(*AUDLVL)
```

5. Ändern Sie die Benutzerprotokollierung, um CL-Commands für die Benutzer aufzuzeichnen, die Sie protokollieren möchten.

```
CHGUSRAUD USRPRF(Benutzerprofil) AUDLVL(*CMD)
```

Wenn Sie wissen möchten, welchen Befehl ein Anwender ausgeführt hat, verwenden Sie den Befehl Display Journal (DSPJRN).

```
DSPJRN JRN(QSYS/QAUDJRN) OUTPUT(*OUTFILE) OUTFILE(xxx/file-name)
```

Die Option *OUTFILE erstellt eine Datei, die die Einträge enthält. Sie können diese Datei abfragen, um die Datensätze für einen bestimmten Anwender und eine bestimmte Zeitperiode abzurufen.

Midrange Computing Redaktion

Querverweise (Cross References)

Wo habe ich diesen Befehl verwendet?

Wenn Sie einen Befehl modifizieren, indem Sie einen Parameter hinzufügen oder löschen oder die Attribute eines Parameters ändern, ist es eventuell erforderlich, einige oder alle CL-Programme zu ändern, die diesen Befehl aufrufen. Aber was tun Sie, wenn Sie sich nicht an die Namen von all diesen CL-Programmen erinnern? Der Befehl RTCMDUSG hilft Ihnen in dieser Situation durch das Spulen einer Liste aller CL-Programme, die den fraglichen Befehl verwenden. Sie können bis zu 50 Commands eingeben, nach denen gesucht werden soll.

Hinweis: Alle HLL-Programme, die Commands durch Aufruf von QCMDEXC aufrufen, werden nicht aufgelistet.

Midrange Computing Redaktion

Fehlernachrichten

Bereinigen des Jobprotokolls

Frage: Ich habe ein CL-Programm entwickelt, das QCMDEXC aufruft, um einen Befehl auszuführen, der als String erstellt wurde. Wenn dieser Befehl einen Fehler aufweist, erhalte ich CPF0001 von QCMDEXC, aber die ursprüngliche CPF0001, die vom Befehl an QCMDEXC gesendet wurde, befindet sich nach wie vor im Jobprotokoll. Ich möchte wirklich ein bereinigtes Jobprotokoll haben, weil ich eine Fehlerbehandlung ausführe. Wie kann ich diese Nachricht, die an QCMDEXC gesendet wird (das natürlich nicht mehr im Aufrufstapel ist) empfangen oder entfernen, wenn ich den Nachrichtenschlüssel nicht kenne und RMVMSG CLEAR(*ALL) oder *ALLINACT nicht verwenden möchte?

Antwort: Verwenden Sie QCAPCMD anstatt von QCMDEXC. Damit wird die reale Ausnahme anstelle der dummen von QCMDEXC zurückgegebenen „Befehl fehlgeschlagen“-Ausnahme zurückgegeben.

Derek Butland

Interaktive Befehlsverarbeitung

Commands, die eine Ausführung in QINTER vermeiden

Einige Commands sollten niemals interaktiv ausgeführt werden. Aber gleichgültig, wie oft man das erzählt, manche Leute glauben immer: „einmal wird schon nichts schaden“. Um diese Mitarbeiter daran zu hindern, bestimmte Commands interaktiv auszuführen, sollte der folgende Befehl für jeden einzuschränkenden Befehl ausgeführt werden:

```
CHGCMD CMD(xxx) ALLOW(*BATCH *BPGM)
```

Wenn jemand versucht, den Befehl interaktiv auszuführen wird eine Fehlermeldung CPD0031-“Command xxx not allowed in this setting“, „Befehl *xxx* ist in dieser Einstellung nicht zulässig“, ausgegeben.

Hier einige Hinweise zur Verwendung dieser Technik: CHGCMD sollte in einem CL-Programm kodiert werden, so dass dieselben Commands auch dann eingegeben werden können, wenn eine neue Version des Betriebssystems installiert wird. Außerdem sollten Sie die Allgemeinheit mit *EXCLUDE davon abhalten, CHGCMD zu verwenden, denn sie könnte diese Änderung einfach rückgängig machen!

Louise Best

Hinweis des Editors: Die beschriebene Technik funktioniert unter der Voraussetzung, dass keine Programme die blockierten Commands legitim ausführen. Um diese Technik zu verbessern, sollte jeder Befehl daher wie folgt geändert werden:

```
CHGCMD CMD(xxx) ALLOW(*BATCH *BPGM *IPGM *EXEC)
```

Wenn Sie REXX verwenden, fügen Sie *BREXX und *IREXX dem Parameter ALLOW hinzu.

Menüs

Ausführung mehrerer Commands von einem SDA-Menü aus, ohne ein CL-Programm zu schreiben

Ich erstelle gerne AS/400-Menüs mit SDA. So kann ich ein Menü rasch und einfach erstellen und mich anschließend anderen Aufgaben zuwenden. Etwas, das mir an SDA-Menüs nicht gefällt, ist die Tatsache, dass ich nur einen Befehl hinter eine Option stellen kann. So muss ich oft ein CL-Programm schreiben, das nur zwei CL-Commands hinter einer Menüoption ausführt, und es gefällt mir nicht, meine Bibliotheken mit winzigen CL-Programmen voll zu stopfen, die nur in einem Menü verwendet werden.

Der Befehl Execute Command (XC) (gezeigt in den Abbildungen 3.3 und 3.4) bietet eine Methode für die Ausführung von mehr als einem Befehl bei der Auswahl einer Menüoption.


```
/*==============================================================*/
/* Kompilierung:                                                */
/*                                                              */
/* CRTCLPGM PGM(XXX/XC) SRCFILE(XXX/QCLSRC)                     */
/*                                                              */
/*==============================================================*/
CMD PROMPT('Ausführung von Commands')

PARM KWD(C) TYPE(*CHAR) LEN(110) MIN(1) +
PROMPT('Commands')

PARM KWD(D) TYPE(*CHAR) LEN(1) DFT('\') +
PROMPT('Befehlsbegrenzer')
```

Abbildung 3.3: Befehl XC bietet eine sehr einfach verwendbare Befehls schnittstelle (Teil 1 von 2).

```
PARM KWD(F) TYPE(*CHAR) LEN(1) RSTD(*YES) DFT(C) +
VALUES(C I) CHOICE('C=cancel, I=ignore') +
PROMPT('Option für fehlgeschlagene Commands')
```

Abbildung 3.3: Befehl XC bietet eine sehr einfach verwendbare Befehls schnittstelle (Teil 2 von 2).


```
/*==================================================================*/
/* Kompilierung:                                                    */
/*                                                                  */
/* CRTCLPGM PGM(XXX/XCL) SRCFILE(XXX/QCLSRC)                        */
/*                                                                  */
/*==================================================================*/
PGM PARM(&CMDSTRING &DELIMITER &FAILOPTION)
DCL VAR(&CMDSTRING) TYPE(*CHAR) LEN(110)
DCL VAR(&DELIMITER) TYPE(*CHAR) LEN(1)
DCL VAR(&FAILOPTION) TYPE(*CHAR) LEN(1)
DCL VAR(&CMDS) TYPE(*CHAR) LEN(111)
DCL VAR(&COMMAND) TYPE(*CHAR) LEN(110)
DCL VAR(&FX) TYPE(*DEC) LEN(3)
DCL VAR(&TX) TYPE(*DEC) LEN(3)
DCL VAR(&MSGDTA) TYPE(*CHAR) LEN(132)
DCL VAR(&MSGF) TYPE(*CHAR) LEN(10)
DCL VAR(&MSGFLIB) TYPE(*CHAR) LEN(10)
DCL VAR(&MSGID) TYPE(*CHAR) LEN(7)
CHGVAR VAR(&CMDS) VALUE(&CMDSTRING *CAT &DELIMITER)
/* Beginn der Extrahierung des nächsten Befehls in der Folge */
NEXTCMD: CHGVAR VAR(&COMMAND) VALUE(' ')
CHGVAR VAR(&TX) VALUE(0)
```

Abbildung 3.4: CL-Programm XCL bietet die Verarbeitungsleistung für den Befehl XC (Teil 1 von 2).

```
/* Erstellung von jeweils einem Befehl 1-Zeichen */
BLDCMD: CHGVAR VAR(&FX) VALUE(&FX + 1)
IF COND(%SST(&CMDS &FX 1) *EQ &DELIMITER) +
THEN(GOTO CMDLBL(RUNCMD))
CHGVAR VAR(&TX) VALUE(&TX + 1)
CHGVAR VAR(%SST(&COMMAND &TX 1)) VALUE(%SST(&CMDS +
&FX 1))
GOTO CMDLBL(BLDCMD)
/* Befehl wurde extrahiert - bitte ausführen */
RUNCMD: IF COND(&COMMAND *NE ' ') THEN(DO)
CALL PGM(QCMDEXC) PARM(&COMMAND 110)
MONMSG MSGID(CPF0000) EXEC(DO)
FWDMSG: RCVMSG MSGDTA(&MSGDTA) MSGID(&MSGID) MSGF(&MSGF) +
SNDMSGFLIB(&MSGFLIB)
IF COND(&MSGID *NE ' ') THEN(DO)
SNDPGMMSG MSGID(&MSGID) MSGF(&MSGFLIB/&MSGF) +
MSGDTA(&MSGDTA)
GOTO CMDLBL(FWDMSG)
ENDDO
IF COND(&FAILOPTION *EQ C) THEN(GOTO +
CMDLBL(ENDPGM))
ENDDO
ENDDO
IF COND(&FX *LT 110) THEN(GOTO CMDLBL(NEXTCMD))
ENDPGM: ENDPGM
```

Abbildung 3.4: CL-Programm XCL bietet die Verarbeitungsleistung für den Befehl XC (Teil 2 von 2).

Der erste Parameter ist eine Reihe von CL-Commands, die von einem Begrenzer voneinander getrennt werden. Das System führt nacheinander jeden Befehl aus. Alle Commands in der Befehlskette müssen durch das Programm QCMDEXC ausgeführt werden können. Um zu prüfen, ob ein Befehl von QCMDEXC ausgeführt werden kann, verwenden Sie den Befehl Display Command (DSPCMD). Wenn der Parameter „Where allowed to run“ *EXEC enthält, können Sie den Befehl innerhalb einer XC-Befehlszeichenkette verwenden. Der zweite Parameter ist der Begrenzer. Standardmäßig wird hierfür

der Querstrich (\) verwendet, aber geeignet ist jedes Zeichen, das nicht im Befehlsstring des Befehls verwendet wird. Der dritte Parameter informiert den Job darüber, was er tun soll, wenn ein Befehl abnormal endet. Sie können den Fehler ignorieren und mit dem nächsten Befehl fortsetzen oder die Verarbeitung der Befehlszeichenkette abbrechen.

Ted Holt

Parameter

Erweiterung der Befehlsparameter-Länge

Hier eine wenig bekannte Midrange-Geschichte, die von Programmierer zu Programmierer weitererzählt wird. Dieses Feature wurde an einem obskuren Ort in den S/38-Handbüchern dokumentiert, und ich bin nicht sicher, dass es in den AS/400-Handbüchern überhaupt erwähnt wird.

Sie können für jede Befehlsbedienerführung ein kommerzielles Pluszeichen (&) gefolgt von einer oder mehreren Leerstellen eingeben, beginnend in der ersten Position eines Eingabefelds, um die Länge zu erweitern. Die maximale Länge beträgt für Bezeichner 80 und für alle anderen Parameter 512.

Um einen Wert einzugeben, der sogar noch länger ist als die maximale Bedienerführungslänge, drücken Sie F3, um in die Anzeige für die Befehlseingabe zu wechseln und den Wert in den Befehls-String einzugeben, ohne die Bedienerführung für Commands zu verwenden. Das kann sehr hilfreich sein, wenn Sie einen Ausdruck eingeben möchten, während Sie ein CL-Programm bearbeiten.

Midrange Computing Redaktion

Positionieren der erforderlichen Parameter nach optionalen Parametern

Wenn Sie schon einmal schriftliche Befehlsdefinitionen erstellt haben, dann haben Sie sicherlich bereits entdeckt, dass mit Erstellung eines Parameters mit einem Standardwert erzwungen wird, dass dieser Parameter optional ist. Sobald ein Parameter als optional definiert ist (MIN-Wert 0) kann kein Parameter, der danach definiert wird, obligatorisch sein (MIN-Wert größer als 0). Dieses Erfordernis führt häufig dazu, dass Programmierer einen Parameter optional werden lassen, wenn er obligatorisch sein sollte. Es gibt jedoch einen Workaround für dieses Problem, den Sie in Betracht ziehen sollten.

Hier ein Beispiel, das diese Technik illustrieren sollte: Angenommen, Sie möchten einen Befehl mit zwei Parametern erstellen. Der erste Parameter ist das FROMDATE, für das Sie einen Standardwert *CURRENT angeben werden. Der zweite Parameter ist TODATE, der angegeben werden muss. Weil FROMDATE optional ist (aufgrund seines Standardwerts), kann TODATE nicht obligatorisch sein.

Um dieses Problem zu umgehen, können Sie eine Anzeigeposition bestimmen, die relativ zu den anderen Parametern im Parameter PROMPT der Anweisung PARM ist. Die Anzeigeposition wird als zweites Element des Parameters PROMPT angegeben – sofort nach dem Bedienerführungstext. Abbildung 3.5 zeigt eine Beispiel, wie das geschehen kann.


```
CMD PROMPT('Beispielbefehl')
  PARM KWD(TODATE) TYPE(*DATE) MIN(1) PROMPT('Bis-Datum' 2)
  PARM KWD(FROMDATE) TYPE(*DATE) DFT(*CURRENT) SPCVAL((*CURRENT 000000)) +
  PROMPT('Von-Datum' 1)
```

Abbildung 3.5: Verwenden Sie diese Beispielbefehls-Source als Richtlinie für das Erlernen der Verwendung der Parameter.

In diesem Beispiel können Sie TODATE zuerst als obligatorischen Parameter definieren, der einen relativen Anzeigepositionswert 2 aufweist. Sie können danach FROMDATE mit einem relativen Anzeigepositionswert 1 definieren. Auf diese Weise erfüllen Sie die Anforderung des Befehls-Compilers, dass obligatorische Parameter vor optionalen Parametern definiert werden müssen. Gleichzeitig sorgen Sie dafür, dass die Parameter in der gewünschten Reihenfolge angezeigt werden.

Robin Klima

Bedienerführung

Farbige Commands

So gut wie jeder weiß, wie Hexadezimal-Codes in den Source-Code eingefügt werden, um die Source einzufärben. Ein Beispiel für hexadezimale Zeichen ist x‘22' für weiß oder High Intensity und x‘38' für rosa. Mit dieser Technik können auch Bedienerführungen farbig gestaltet werden.

Bei der Erstellung der Source für Ihren Befehl positionieren Sie ein hexadezimales Zeichen wie z.B. x‘38' als erstes Byte in den PROMPT-Parameter. Nach Kompilierung des Befehls verfügt der Befehl über eine farbige Bedienerführung. Der in Abbildung 3.6 gezeigte Beispiel-Code zeigt, wo der Hexadezimal-Code (angezeigt durch x) positioniert werden soll, um Ihren Befehl einzufärben.


```
CMD PROMPT('Befehlstitel')
PARM KWD(FLD) TYPE(*CHAR) LEN(10) +
     PROMPT(x'38')
```

Abbildung 3.6: Mit diesem Befehl färben Sie Ihre Source ein!

Todd Fisher

Shortcut-Commands

Abgekürzte OS/400 Commands

Eine Eigenschaft von OS/400, die dieses Betriebssystem bedienerfreundlicher als andere macht, ist die logische Namensgebung der Commands. Häufig kann der Namen eines Befehls einfach erraten werden. Der Nachteil davon ist jedoch, dass die Befehlsnamen oft sehr lang sind.

Sie können sich täglich Hunderte von Tastenanschlägen sparen, indem Sie abgekürzte Versionen der von Ihnen häufig verwendeten OS/400-Commands erstellen. Ich werde Ihnen zwei meiner bevorzugten Abkürzungen vorstellen: SJ und WJ4.

Der Befehl SJ ist eine abgekürzte Form des Befehls Work with Submitted Jobs (WRKSBMJOB) unter Verwendung des Parameters SBMFROM(*JOB). Ich mag SJ weil es mir erlaubt, nur die Jobs zu sehen, die vom aktuellen Job übergeben wurden, und weil SJ mindestens 12 Tastenanschläge kürzer ist als sein natives Gegenstück. Die Source dieses Befehls wird in Abbildung 3.7 gezeigt.


```
/*==============================================================*/
/* Kompilierung:                                                */
/*                                                              */
/* CRTCMD CMD(XXX/SJ) PGM(XXX/QCMDEXC) +                        */
/* SRCFILE(XXX/QCMDSRC)                                         */
/*                                                              */
/*==============================================================*/
```

Abbildung 3.7: Der abgekürzte WRKSBMJOB-Befehl SJ erleichtert die Arbeit mit übergebenen Jobs (Teil 1 von 2).

```
CMD PROMPT('WRKSBMJOB *JOB')
PARM KWD(CMD) TYPE(*CHAR) LEN(23) +
CONSTANT('WRKSBMJOB SBMFROM(*JOB)')
PARM KWD(LEN) TYPE(*DEC) LEN(15 5) CONSTANT(23)
```

Abbildung 3.7: Der abgekürzte WRKSBMJOB-Befehl SJ erleichtert die Arbeit mit übergebenen Jobs (Teil 2 von 2).

Der Befehl WJ4 erlaubt Ihnen, nur die Spool-Dateien zu sehen, die vom aktuellen Job erstellt wurden. WJ4 ähnelt der Eingabe des Befehls Work with Job (WRKJOB) mit Option 4. Sie müssen nicht Seite um Seite der Spool-Dateien abarbeiten, wie es bei WRKOUTQ oder WRKSPLF erforderlich ist. Die Befehls-Source WJ4 wird in Abbildung 3.8 gezeigt.


```
/*==============================================================*/
/* Kompilierung:                                                */
/*                                                              */
/* CRTCMD CMD(XXX/WJ4) PGM(XXX/QCMDEXC) +                       */
/* SRCFILE(XXX/QCMDSRC)                                         */
/*                                                              */
/*==============================================================*/
CMD PROMPT('WRKJOB Option 4')
PARM KWD(CMD) TYPE(*CHAR) LEN(20) +
CONSTANT('WRKJOB OPTION(*SPLF)')
PARM KWD(LEN) TYPE(*DEC) LEN(15 5) CONSTANT(20)
```

Abbildung 3.8: Der abgekürzte WRKJOB-Befehl WJ4 erleichtert die Arbeit mit gespoolten Dateien eines Jobs.

In beiden diesen Beispielen habe ich QCMDEXC als befehlsverarbeitendes Programm angegeben, das mir das Schreiben eines CPP erspart. Ich wette, es gibt viele andere AS/400-Commands, die Sie häufig verwenden, die so abgekürzt werden können und Ihnen damit täglich viele Tastenanschläge ersparen.

Ted Holt

AS/400-Kurzbefehle

Zu den angenehmen Seiten des System/36 gehört die Funktionaliät für die Eingabe von Abkürzungen für häufig verwendete Commands: D U für die Anzeige von Benutzers, D P für die Anzeige von Spool-Dateien, D W für die Anzeige von Arbeitsstationen usw.

Auf der AS/400 (die in einer System/36-Umgebung ausgeführt wird) funktionieren diese Commands in etwa genauso wie auf der S/36. Im nativen Modus sind derartige Kurzbefehle nicht verfügbar. Die AS/400 gibt uns jedoch die Tools an die Hand, um unsere eigenen abgekürzten Commands zu erstellen. Befehl und Programm (Abbildungen 3.9 und 3.10) zeigen, wie einige der häufig verwendeten „Arbeiten mit“-Commands abgekürzt werden können.

```
CMD       PROMPT('Mit ausgewählten Jobs arbeiten')

PARM      KWD(JOB) TYPE(*CHAR) LEN(2) RSTD(*YES) +
                VALUES('A' 'C' 'D' 'JL' 'JQ' 'P' 'P2' 'Q' +
                'QP' 'S' 'SJ' 'U' 'W') +
                CHOICE(*VALUES) +
                PROMPT('Auswahl des Jobs für die Arbeit')
```

Abbildung 3.9: Mit ausgewählten Jobs zu arbeiten ist mit diesem Befehl ganz einfach.

```
   PGM PARM(&OPT)

   DCL VAR(&OPT) TYPE(*CHAR) LEN(2)

   IF COND(&OPT *EQ 'A') THEN(WRKACTJOB)
   IF COND(&OPT *EQ 'C') THEN(WRKCFGSTS CFGTYPE(*DEV))
   IF COND(&OPT *EQ 'D') THEN(WRKDOC)
   IF COND(&OPT *EQ 'JL') THEN(WRKOUTQ OUTQ(JOBLOGS))
   IF COND(&OPT *EQ 'JQ') THEN(WRKJOBQ JOBQ(QBATCH))
   IF COND(&OPT *EQ 'P') THEN(WRKOUTQ OUTQ(PRT01))
   IF COND(&OPT *EQ 'P2') THEN(WRKOUTQ OUTQ(PRT02))
   IF COND(&OPT *EQ 'Q') THEN(WRKOUTQ)
   IF COND(&OPT *EQ 'QP') THEN(WRKOUTQ OUTQ(QPRINT))
   IF COND(&OPT *EQ 'S') THEN(WRKSPLF)
   IF COND(&OPT *EQ 'SJ') THEN(WRKSBMJOB)
   IF COND(&OPT *EQ 'U') THEN(WRKUSRJOB)
   IF COND(&OPT *EQ 'W') THEN(WRKWTR)
   IF COND(&OPT *EQ ' ') THEN(WRKJOB)

ENDPGM: +
   ENDPGM
```

Abbildung 3.10: Der CL-Treiber für den Befehl Work With Selected Jobs bietet die Leistungsfähigkeit, die hinter dem Befehl steht.

Der Befehl wird einfach als „W“ bezeichnet und wird den Befehl Work with Job (WRKJOB) ausführen. Fügen Sie dem Programm, das in den Abbildungen 3.9 und 3.10 gezeigt wird, Ihren eigenen Kurzbefehl hinzu. Erstellen Sie einen völlig neuen Befehl, wie z.B. den Befehl „C“ für die häufig verwendeten AS/400-“Change“-Commands. Vielleicht möchten Sie auch die Kurzbefehle im Stil von System/36 im nativen Modus beibehalten. Gleichgültig was Sie tun, Sie werden sich ziemlich viel Tipparbeit sparen.

Steven Kontos

Generische Commands und CPPs

Häufig müssen Sie einen Ad-hoc-Befehl ohne Parameter schreiben, um eine sehr spezielle Aufgabe wiederholt auszuführen. Angenommen, Sie müssen wiederholt feststellen, welche Benutzer an allen Anzeigestationen angemeldet sind. Sie wissen, dass der Befehl Work with User Jobs (WRKUSRJOB) ihnen diese Informationen bereitstellt, aber Sie müssen dafür einen sehr langen Befehl eingeben:

```
WRKUSRJOB USER(*ALL) STATUS(*ACTIVE) JOBTYPE(*INTERACT)
```

Wenn Sie blitzschnell Schreibmaschine schreiben können (sagen wir einmal mit 120 Wörtern pro Minute) macht es Ihnen vielleicht nichts aus, das Dutzend Mal jeden Tag einzutippen. Aber das einfache Volk bevorzugt eine weniger komplizierte Methode – wie z.B. die Eingabe von DSPSGNUSR (das für Display Signed-on Users steht, ein Befehlsnamen, den ich erfunden habe) und Drücken der Eingabetaste. Der Befehl DSPSGNUSR wird natürlich ein Befehlsverarbeitungsprogramm bzw. CPP benötigen, was in den meisten Fällen bedeutet, dass ein CL-Source-Code geschrieben und kompiliert werden muss. Wie Sie sich vorstellen können, werden nicht nur die Commands sondern auch die CPPs auf der Festplatte überhand nehmen.

Angesichts dieser Tatsachen beschloss ich, ein generisches CPP (dem ich einfach den Namen GENCPP gab) für die Verwendung in diesen Fällen zu erstellen. Zentraler Gedanke ist dabei, dass der Befehl nur einen Parameter an das CPP übergibt: eine Nachrichten-ID. Wenn das CPP diesen Parameter empfängt, ruft das CPP den entsprechenden Nachrichtentext aus einer bestimmten Nachrichtendatei ab und verwendet den Nachrichtentext als Befehls-String, der mit QCMDEXC ausgeführt wird. Abbildung 3.11 zeigt die Befehls-Source für DSPSGNUSR. Wie Sie sehen können, zeigt sein einziger Parameter MSGID einen konstanten Wert von CMD0001. Da der Parameter eine Konstante ist, wird er nie gezeigt, wenn der Benutzer DSPSGNUSR aufruft, und der Benutzer kann seinen Wert in keinerlei Weise überschreiben.

Wenn die CPP mit der Ausführung beginnt (Abbildung 3.12), erhält sie die Nachrichten-ID und verwendet den Befehl Retrieve Message (RTVMSG), um den

Text von der Nachrichtenbeschreibung in der Nachrichtendatei GENCPPMSGF abzurufen, den Sie selbst erstellen und pflegen müssen. Anschließend sendet die CPP eine *INFO- Nachricht an sich selbst, bevor der Befehls-String ausgeführt wird. Der Grund für die *INFO-Nachricht ist, dass etwas im Jobprotokoll sein sollte, das den auszuführenden Befehl identifiziert. Wenn Sie diese Nachricht nicht senden, zeichnet das Jobprotokoll einen CALL von QCMDEXC auf, aber Sie erhalten keine Informationen darüber, was ausgeführt wird.

```
/*==================================================================*/
/* Kompilierung:                                                    */
/*                                                                  */
/* CRTCMD CMD(XXX/DSPSGNUSR) PGM(XXX/GENCPP) +                      */
/* SRCFILE(XXX/QCMDSRC) TEXT('Anzeige +                             */
/* Angemeldete Benutzer')                                           */
/*                                                                  */
/*==================================================================*/
CMD PROMPT('Anzeige der angemeldeten Benutzer')
PARM KWD(MSGID) TYPE(*NAME) LEN(7) CONSTANT(CMD0001)
```

Abbildung 3.11: Der Befehl DSPSGNUSR informiert Sie darüber, welche Benutzer wo angemeldet sind.

```
/*==================================================================*/
/* Kompilierung:                                                    */
/*                                                                  */
/* CRTCLPGM PGM(XXX/GENCPP) SRCFILE(XXX/QCLSRC) +                   */
/* TEXT('Generische Befehls-CPP')                                   */
/*                                                                  */
/*==================================================================*/
```

Abbildung 3.12: Diese generische CPP erhält eine Nachrichten-ID (Teil 1 von 2).

```
PGM PARM(&IN_MSGID)
DCL VAR(&CMDSTR) TYPE(*CHAR) LEN(256)
DCL VAR(&CMDSTRLEN) TYPE(*DEC) LEN(15 5)
DCL VAR(&IN_MSGID) TYPE(*CHAR) LEN(7)
DCL VAR(&MSGDTA) TYPE(*CHAR) LEN(256)
DCL VAR(&MSGF) TYPE(*CHAR) LEN(10)
DCL VAR(&MSGFLIB) TYPE(*CHAR) LEN(10)
DCL VAR(&MSGID) TYPE(*CHAR) LEN(7)
DCL VAR(&MSGTXT) TYPE(*CHAR) LEN(256)
DCL VAR(&MSGTXTLEN) TYPE(*DEC) LEN(5)
MONMSG MSGID(CPF0000 MCH0000) EXEC(GOTO CMDLBL(ERROR))
/* Abruf des Nachrichtentexts und Verwendung als Befehlsstring */
RTVMSG MSGID(&IN_MSGID) MSGF(GENCPPMSGF) +
MSG(&MSGTXT) MSGLEN(&MSGTXTLEN)
CHGVAR VAR(&CMDSTR) VALUE(%SST(&MSGTXT 1 &MSGTXTLEN))
CHGVAR VAR(&CMDSTRLEN) VALUE(&MSGTXTLEN)
/* Ausführung des Befehls */
SNDPGMMSG MSG('Executing command' *BCAT &CMDSTR) +
TOPGMQ(*SAME (*)) MSGTYPE(*INFO)
CALL PGM(QCMDEXC) PARM(&CMDSTR &CMDSTRLEN)
RETURN
/* Weiterleitung aller Fehlernachrichten an das aufrufende Programm */
ERROR:
RCVMSG MSGTYPE(*EXCP) MSGDTA(&MSGDTA) MSGID(&MSGID) +
MSGF(&MSGF) SNDMSGFLIB(&MSGFLIB)
MONMSG MSGID(CPF0000)
SNDPGMMSG MSGID(&MSGID) MSGF(&MSGFLIB/&MSGF) +
MSGDTA(&MSGDTA) MSGTYPE(*ESCAPE)
MONMSG MSGID(CPF0000)
ENDPGM
```

Abbildung 3.12: Diese generische CPP erhält eine Nachrichten-ID (Teil 2 von 2).

Um diese Technik zu implementieren, benötigen Sie irgendwo in Ihrer Bibliotheksliste eine Nachrichtendatei mit der Bezeichnung GENCPPMSGF. Wenn Sie es vorziehen, einen anderen Nachrichtendateinamen zu verwenden, sollten Sie das auch tun; vergessen Sie aber nicht, den Nachrichten-

dateinamen, auf den verwiesen wird, in der RTVMSG-Anweisung der CPP zu ändern. Es müssen auch Nachrichtenbeschreibungen vorhanden sein (eine pro generischem Befehl). Ich habe z.B. DSPSGNUSR kodiert, so dass eine Nachrichtenbeschreibung CMD0001 verwendet wird. Anschließend muss ich den Befehl Add Message Description (ADDMSGD) ausführen, um diese Nachrichten-ID in der ausgewählten Nachrichtendatei zu erstellen. Ihr Code sollte so aussehen:

```
CRTMSGF  xxx/GENCPPMSGF ADDMSGD CMD0001 xxx/GENCPPMSGF +
         MSG('WRKUSRJOB USER(*ALL) STATUS(*ACTIVE) +
         JOBTYPE(*INTERACT)') SECLVL('Anzeige aller angemeldeten Benutzer')
```

Wenn Sie einen weiteren generischen Befehl erstellen, verwenden Sie DSPSGNUSR als Modell, ändern den Text in der Anweisung CMD und die Nachrichten-ID im Parameter MSGID. Anschließend führen Sie ADDMSGD aus, um die neue Nachrichtenbeschreibung zu erstellen und – voilà! – der neue Befehl kann verwendet werden.

Ernie Malaga

UIM

Unterstützung für Hilfemasken

Frage: Ich versuche, in meinen neuen Commands auf einen vorhandenen IBM-Maskengruppen-Hilfetext zu verweisen. Anstatt die User Interface Manager-Daten (UIM) für die Hilfemaske erneut einzugeben, möchte ich eine Verknüpfung mit einer vorhandenen IBM-Hilfemaske mit aufnehmen. Wie finde ich die Namen der bereits vorhandenen Hilfemaskengruppen heraus, damit ich sie wiederverwenden kann?

Antwort: Versuchen Sie, den Befehl Display Command (DSPCMD) zu verwenden, um die Informationen aufzufinden, nach denen Sie suchen. Sie können z.B. DSPCMD verwenden und dabei den Parameterwert von WRKUSRJOB angeben, um Befehlsinformationen für den Befehl Work with User Jobs (WRKUSRJOB) anzuzeigen. Sobald die Informationen angezeigt werden, blättern Sie zur zweiten Seite. In der Mitte des Bildschirms finden Sie den Namen der Hilfemaske.

Vadim Rozen

Mit Commands arbeiten

Verwendung von generischen Commands in der Befehlszeile

Wussten Sie, dass Sie, wenn Sie z.B. in der Befehlszeile WRK* eingeben und die Eingabetaste drücken, alle Commands erhalten, die mit WRK beginnen und den gewünschten Befehl auswählen können? Sie können auch WRKU* oder WRKUSR* eingeben, wenn Sie wissen, wie der Befehl beginnt, aber vergessen haben, wie der Rest des Befehls lautet. Diese Listen sind auf den Befehl Select Command (SLTCMD) zurückzuführen, der automatisch aktiviert wird.

Weil ich in der Regel noch genug vom ersten Teil eines Befehls weiß, um die Möglichkeiten etwas einzuschränken, ziehe ich diese Methode der Verwendung von F4 oder den GO CM*Dxxx*-Menüs vor. Diese Methode erspart mir das Durchsuchen aller Menüs. Ich habe das eines Tages durch Zufall erfahren, während ich mit einem schrecklich langsamen Emulationspaket arbeitete. Ich gab eilig einen Befehl ein, vertippte mich, und gab dabei versehentlich einen Befehl ein, der mit einem Sternchen endete.

Rebecca Whittemore

Hinweis des Editors: Die direkte Verwendung des Befehls SLTCMD ist riskant. Auf den üblichen QWERTY-Tastaturen liegen die Tasten S und D nebeneinander. Sie könnten versehentlich DLTCMD statt SLTCMD eingeben. Weil der Befehl Delete Command (DLTCMD) einen generischen Befehlsnamen akzeptiert, ist es möglich, versehentlich eine Reihe der von IBM gelieferten Commands zu löschen. Aus diesem Grund wird von der Verwendung von SLTCMD abgeraten.